MASTER IN COMPUTER SCIENCE

A rectification-based approach to panoramic generation and depth estimation in computer vision

Stouffs, Julien; Willame, Yannick

Award date:
2003

Facultés Universitaires Notre-Dame de la Paix
Namur
Institut d'Informatique

A rectification-based approach to panoramic generation and depth estimation in computer vision

Julien Stouffs et Yannick Willame

Mémoire présenté en vue
de l'obtention du grade
de Maître en Informatique

Année académique 2002-2003

Abstract The collection of accurate information in an environment destroyed by a disaster is a paramount task in rescue activities. Indeed, a visualisation of the state of decay and a localisation of the possible victims are often preferable before a human intervention. In order to obtain relevant information, a total sight of the scene is planned. With this end in view, a module equipped with several cameras is developed in such manner to cover the entire field. Our work contributes to the development of two algorithms based on the rectification of images, one generating a panoramic view and the other estimating the distance measurement. Thus, the first part of this thesis presents a non exhaustive overview of the theoretical principles of the computer vision. This state of the art constitutes a solid base for the comprehension of the research field. Then, the second part defines in detail the different stages of the two algorithms, illustrating the suggested fitting of the various theoretical tools.

Keywords Computer vision, stereovision, image registration, image rectification, edge detection, depth estimation, panoramic view,

Résumé La collecte d'informations précises dans un environnement détruit par un désastre est une tâche primordiale dans les activités de secours. En effet, une visualisation de l'état de délabrement et une localisation des éventuelles victimes sont souvent préférables avant une intervention humaine. En vue d'obtenir des informations pertinentes, une vue globale de la scène est envisagée. Dans cet objectif, un module équipé de plusieurs caméras est développé de telle manière à couvrir l'entièreté du milieu. Notre travail contribue à l'élaboration de deux algorithmes basés sur la rectification d'images, l'un générant une vue panoramique et l'autre estimant les distances. Ainsi, la première partie de ce mémoire présente une récapitulation non exhaustive des principes théoriques de la vision par ordinateur. Cet état de l'art constitue une base solide pour la compréhension du domaine de recherche. Ensuite, la seconde partie définit en détail les différentes étapes des deux algorithmes, illustrant l'agencement proposé des divers outils théoriques.

Mots-clés Vision par ordinateur, stéréovision, coregistration d'images, rectification d'image, détection de contours, estimation de la profondeur, vision panoramique

We wish to thank our professor Pierre-Yves Schobbens for the autonomy that he gave us, his attentive and relevant reading and his judicious advices.

We also wish to thank the professor Satoshi Tadokoro for his welcome, his extreme availability, his advised orientation for our work and especially for all the facilities during our stay in Japan.

Furthermore, we wish to thank the two above mentioned professors for the freedom for research that they granted to us during our training and during the drafting of our thesis.

Obviously, we would like to thank all the ***International Rescue System Institute*** *Kobe laboratory members for their friendship and their welcome, and in particular Takumi Kishima for his help and his availability. We do not forget either the "Welcome" parties and the end of year party. We also have a good memory of the moments spent with Takuma Tanaka and Shiro Takashima.*

We are grateful towards our parents for their moral and financial support during these years of studies and towards our girlfriends for their presence and their encouragements.

At last, we give thanks to all our friends with whom we spent a lot of great moments during our cursus.

Contents

Introduction

Introduction

Our work is inscribed in the project «*An Intelligent Sensor Head for Information Collection in Debris*»[TKSW03] whose goal is to provide automated help for rescue activities, typically after an earthquake.

Great advantages can be taken in robotic solutions and automatic image processing for rescue activities. Information collection in debris (like victim information, structural information,...) is one of the most important tasks in large-scale disasters like the 1995 Kobe Earthquake or the 2001 WTC attack. Camera systems mounted on robots could provide great help in search and rescue operations.

This project is to develop an intelligent sensor head offering a wide view in order to collect efficient and thorough information. To obtain omnidirectional images horizontally and vertically, a number of cameras equips the sensor head forming what we call a *compound eye camera*.

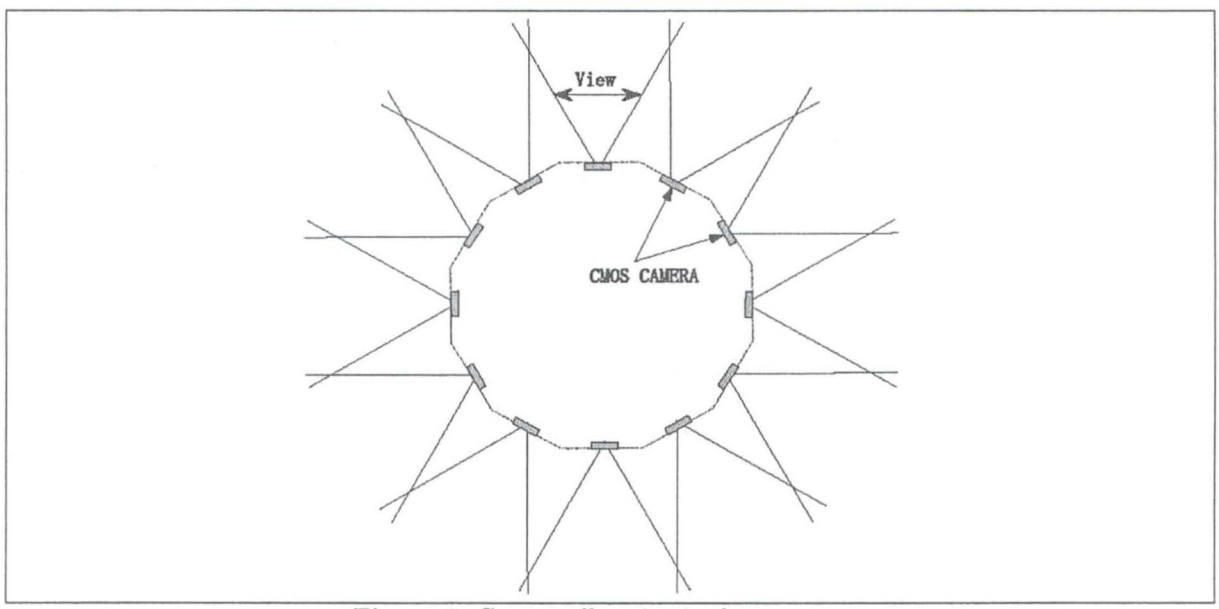


Figure 1: Camera allocation in the prototype.

The first stage is the development of a prototype which has 12 cameras only horizontally as in figure 1. Thus, the angle between two cameras is of 30 °. This configuration is justified by the picture angles of the cameras which are of 58.5 ° horizontally and of 47.5 ° vertically. In this way, the images overlap and a panoramic image can be composed from several source images.

In addition to panoramic view, it is possible to compute 3D position of points in the scene using binocular stereopsis. Furthermore, the proposed method allows human to easily recognise the scene in detail including depth estimation.

- a visual navigation inside the panoramic view;
- a 3D mapping based on distance measurement.

The main contribution of this work is to offer an overview of the theoretical basis and the most important principles in computer vision. Our approach from scratch facilitates the comprehension of the major concepts. Thus, anyone can quickly understand the topic of this research field. Then our implementation allow to estimate the efficiency of our proposed method. We published it in [TKSW03].

To introduce the first part, the chapter one simply recalls how images can be related through various 2D geometric transformations. The chapter two intuitively explains the camera model and the rendering of digital images. Next the translation of this model is done in mathematical form. In this way, the epipolar geometry highlights the relation between two images in a stereoscopic system. In this chapter, the core of our approach, namely the *rectification* algorithm, is also presented. The idea is to align images coming from different viewpoints by suppressing the angle between them. Then the chapter three presents the most used registration methods. The objective of image registration is to find corresponding points between images. It is known to be one of the most difficult task in computer vision when accurate results are needed. The chapter four denotes how to simply recover a 3D point from corresponding points in two images. In the chapter five, the most known edge detectors are reviewed and finally the most used one - the *Canny edge detector*- is detailed. It is an important step because it reduces the data volume while keeping the relevant information contained in an image.

At last, the second part explains the proposed solutions for the panoramic generation and the depth estimation according to the principles and concepts introduced in the first part. The main stages of the two algorithms are shown on a flowchart and each of them is detailed, illustrating the theoretical tools defined in the state of the art.

Part I

State of the art

Chapter 1

Geometric transformations

Introduction

Above all, it is necessary to recall the different 2D geometric transformations that can relate images [Fis02, Aga02, Lev95, FLP01]. In what follows, each type of transformations is linear and invertible. Therefore, if the matrix $\mathcal{M}$ transforms $\vec{p}$ to $\vec{p}\,'$ (i.e. $\vec{p}\,' = \mathcal{M}\vec{p}$), then the inverse matrix $\mathcal{M}^{-1}$ transforms $\vec{p}\,'$ back to $\vec{p}$ (i.e. $\vec{p} = \mathcal{M}^{-1}\vec{p}\,'$). So the product of $\mathcal{M}$ and $\mathcal{M}^{-1}$ is the identity matrix i.e. no transformation is applied.

In this chapter, the homogeneous coordinates system is briefly recalled. Then, the most used 2D geometric transformations are reviewed in the order of their degree of freedom. Hence each transformation is a subset of the next one and more general type of transformations means weaker invariants.

1.1 Homogeneous coordinates

Working with geometric transformations and geometric handling involves working in homogeneous coordinates. That is, a 2D point P with Cartesian coordinates (x, y) becomes a column vector whose third component is set to 1 i.e. $\vec{p} = (x, y, 1)^T$. So the relationship between $\vec{p}$ and $\vec{p}\,'$ can be expressed in matrix notation. Generally, a vector of three real numbers $(x, y, z)^T$ with $z \neq 0$ is the homogeneous coordinates for the point P with Cartesian coordinates $(x/z, y/z)$ and P has many homogeneous coordinates[1]. Indeed, $(x', y', z')^T$ represents the same point P as $(x, y, z)^T$ if $(x', y', z')^T$ is a scalar multiple of $(x, y, z)^T$ i.e.

$$(x', y', z')^T = (\lambda x, \lambda y, \lambda z)^T \tag{1.1}$$

where λ is a scalar, $\lambda \neq 0$.

More generally, a point in a n-dimensional space will be represented as a vector of size $n + 1$.

1.2 Euclidean transformation

The basic geometric transformations are Euclidean transformations or isometries. Isometry means that the distance between any two coordinate locations remains unchanged by the transformation. It concerns:

rotations which change orientation about the coordinate origin;

translations which change position about the coordinate origin.

Note that reflections are also an isometry.

In this case, we are in the Euclidean world coordinates system. Putting these two basic transformations together defines an Euclidean (rigid body) transformation which has the following characteristics:

- straight lines remain straight;

- parallel lines remain parallel;
- shapes are preserved.

Thus lengths, angles and areas are invariant.

The mathematical notation in homogeneous coordinates allows us to combine translation and rotation into a single matrix as follows:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & \Delta_x \\ \sin\theta & \cos\theta & \Delta_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{1.2}$$

The transformation matrix is a 3×3 matrix that depends on three degrees of freedom which are θ for the rotation and Δ_x, Δ_y for the translation along the x-axis and the y-axis respectively.

1.3 Similarity transformation

Another basic transformation is scaling which stretches or shrinks the lengths i.e. the real world Euclidean distance between any two coordinate locations to be multiplied by a real number λ (the scaling factor). A proportional scaling transformation centred at the origin with an isometry forms a similarity transformation or homothety. Adding λ to the isometry matrix defines the similarity transformation and the matrix is now:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \lambda\cos\theta & -\lambda\sin\theta & \Delta_x \\ \lambda\sin\theta & \lambda\cos\theta & \Delta_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{1.3}$$

With these four degrees of freedom (λ, θ, Δ_x and Δ_y), the angles and shapes are left unchanged and the lengths and areas may change in a proportional way.

1.4 Affine transformation

In an affine transformation, the x and y dimensions can be scaled or sheared[2] in a non proportional way. Affine transformations are linear in the sense that they map straight lines into straight lines. They also keep the ratio of lengths on parallel lines and, such as similarity transformations, they preserve the parallelism of lines and the ratio of areas. The transformation matrix is still a 3×3 matrix, but with six degrees of freedom. The equation is commonly written as:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{1.4}$$

where

- m_0 and m_4 change the scale independently;

- m_1 and m_3 are the shearing factors;
- m_2 and m_5 represent the translation.

Note that, as for other transformations, the transformation matrix is required to be nonsingular.

In short, affine transformations do not preserve lengths and angles and thus change the shape of geometric objects.

1.5 Projective transformation

A projective transformation, also called homography, is an invertible mapping such that collinear points remain collinear[3] after transformation, i.e. straight lines remain straight and lines are mapped to lines (but the parallelism is not necessarily preserved). So with projective transformations, collinearity and cross ratio[4] along lines are invariants. The 3×3 matrix is defined up to a non-zero scale factor[5] and has eight degrees of freedom as follows:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{1.5}$$

Here, m_6 and $m7$ influence the parallelism of lines.

Projective transformations can map points at infinity to finite points called vanishing points whereas affine transformations map them to points at infinity.

1.6 Overview

A similarity transformation is sufficient to match two images of a planar scene taken from the same viewing angle but from different positions neglecting the stereoscopic effect (i.e. assuming that the distance of objects is large). That is, the camera can rotate about its optical axis[6] or the camera can move but the optical axis should remain parallel. A projective transformation accounts for distortions which occur when images of a 3D static scene are taken at fixed locations, or when multiple images of a planar scene (e.g. table, painting,...) are taken from arbitrary locations. In practice, the projective model is also a good approximation for a 3D scene when the movement of the camera is small compared to its depth, and for a scene when the relative distances between objects in the scene is small compared to its depth.

The two following tables give an overview of the different geometric transformations and allow us to identify the differences between them and their effects.

[3] A set of points is collinear if they all lie on the same straight line.

[4] If we have four points A, B, C and D lying on a straight line, then the ratio $\frac{AB.CD}{AD.CB}$ is called the cross ratio.

[5] A projective transformation λA (with $\lambda \neq 0$) is the same as A since they map to projectively equivalent

	Euclidean	similarity	affine	projective
Transformations				
rotation, translation	√	√	√	√
isotropic scaling		√	√	√
scaling along axes, shear			√	√
perspective projections				√
Invariants				
distance	√			
angles, ratios of distances	√	√		
parallelism, center of mass, ratio of areas	√	√	√	
incidence[5], cross-ratio	√	√	√	√

Table 1.1: This table [FLP01] shows an ordering of geometries : particular transformations and properties left invariant by the transformations. Each geometry is a subset of the next. More general transformations mean weaker invariants.

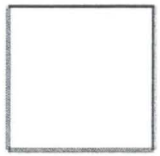
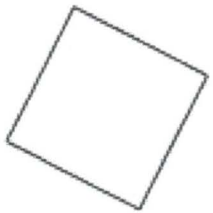
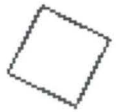
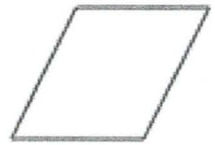
	Euclidean	Similarity	Affine	
Matrix $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} =$	$\begin{pmatrix} \cos\theta & -\sin\theta & \Delta_x \\ \sin\theta & \cos\theta & \Delta_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$	$\begin{pmatrix} \lambda\cos\theta & -\lambda\sin\theta & \Delta_x \\ \lambda\sin\theta & \lambda\cos\theta & \Delta_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$	$\begin{pmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$	$\begin{pmatrix} m \\ m \\ m \end{pmatrix}$
degrees of freedom	3	4	6	
Square 				

Table 1.2: Common geometric transformations that can relate two images. Homogeneous coordinates are used to allow to express the transformations in t of matrix multiplications.

Chapter 2

Imaging and camera model

2.1 Computer vision

The aim of computer vision is the extraction of tridimensional information (or in a general way, all that human being can do with his own visual system) from a set of images captured with an artificial system (such an electronic camera). Without any constraint, it seems difficult to be able to find the information of the surrounding world from a set of bidimensional projections. However, the biological systems provide a proof of the possibility of using vision for problems such navigation, object recognition, scene analysis, depth calculation, ...

Thus, a logical way to fit the problem of vision in artificial system is trying to copy the biological vision. All this section is greatly inspired from [Fus98].

2.1.1 Perspective projection

The simplest geometrical model of imaging is the *pinhole camera.* The pinhole model can be seen as a box with a hole in one of its side. The light rays will be projected to the back of the box and draw an inverted and scaled image of the object. (See figure 2.1)

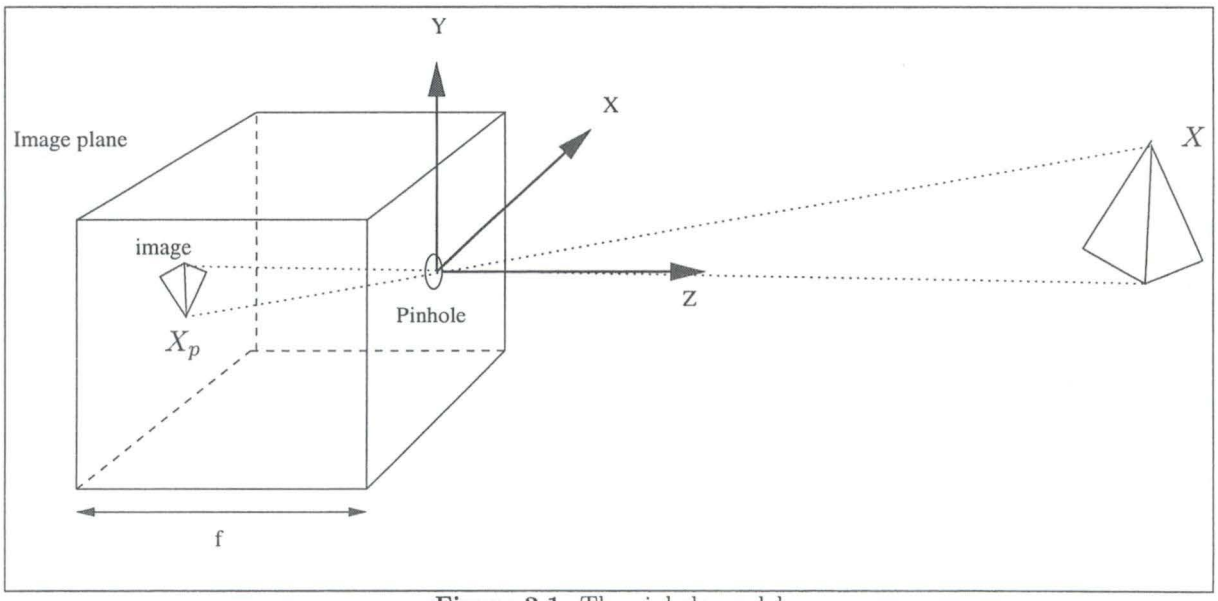


Figure 2.1: The pinhole model

This model can be expressed in a mathematical way. Let X be a point in the scene with coordinates (X, Y, Z) and X_p its projection on the image plane with coordinates (X', Y', Z'). If f is the distance between the pinhole and the image plane, by similar triangle, we can derive the following equations:

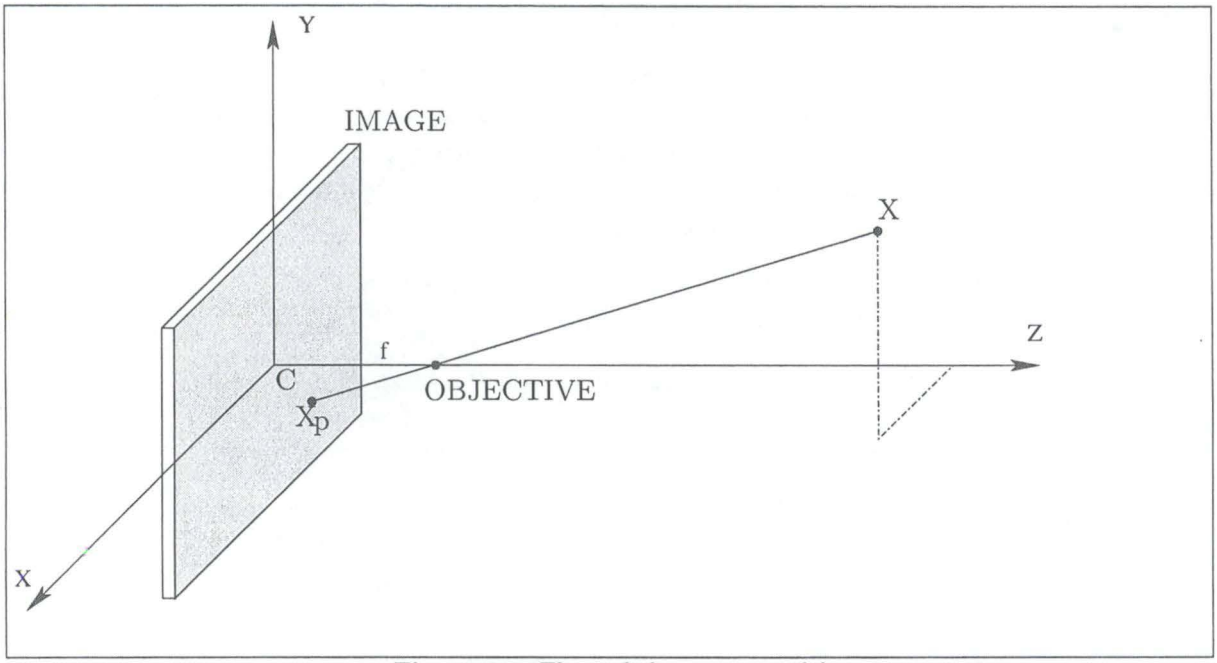


Figure 2.2: The pinhole camera model

Thus, we obtain

$$\begin{aligned} X' &= \frac{-fX}{Z} \\ Y' &= \frac{-fY}{Z} \\ Z' &= -f \end{aligned} \tag{2.2}$$

These equations describe the *perspective projection* or *central projection* which defines the image formation process. Note that the process is not linear because of the division by Z and that the image is inverted left-right and up-down as specified in the equations by the negative signs. Equivalently, we can imagine to put the image plane in front of the pinhole. Then we obtain a non-inverted image and this will simplify the computation since the negative signs will disappear from the equations.

2.1.2 Optics

In the pinhole model, for each scene point, there is only one light ray which reaches the image plane. But in the real case, the lens is actually wider than a pinhole to collect more light. The main drawback is that only a part of the scene can be in sharp focus in the same time. Thus, a common way to approximate a complex optical system is to use a *thin lens*. A thin lens has the basic following properties (see figure 2.3):

1. Any ray entering the lens parallel to the optical axis on one side goes through the *focus* F on the other side;
2. Any ray going through the *lens centre* C is not deflected.

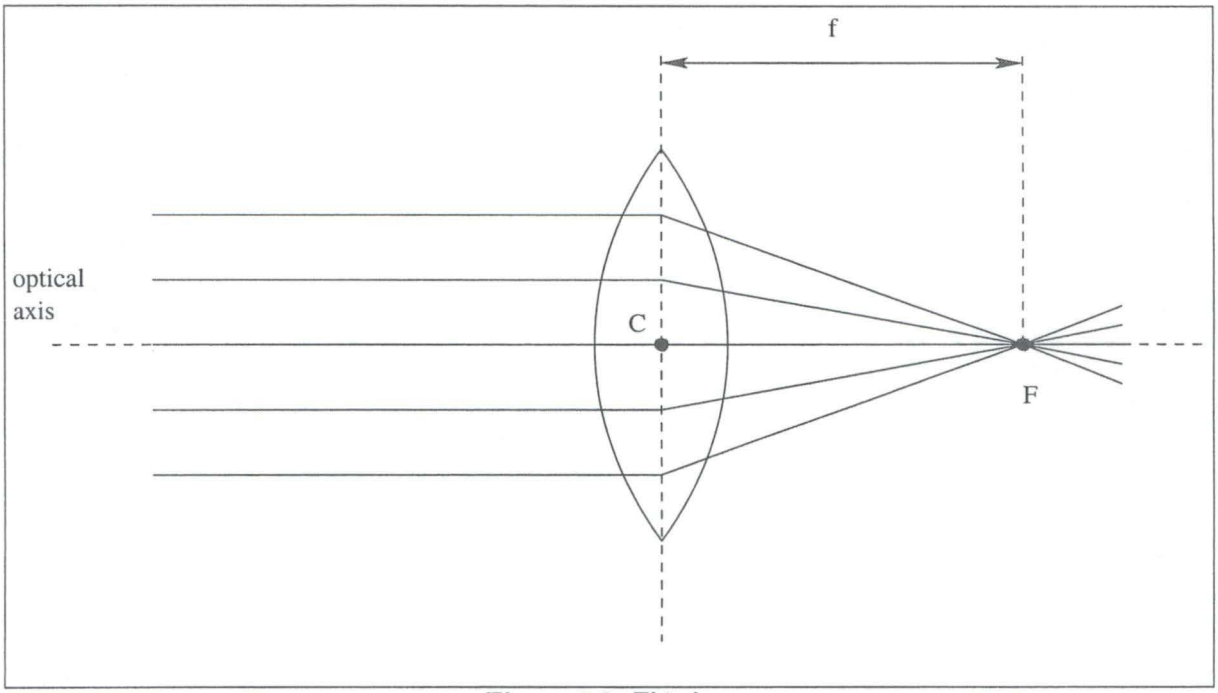


Figure 2.3: Thin lens

The distance f between the focus F and the lens centre C is the *focal length*, it depends on the curvature of both sides of the lens and on refraction index of the material.

Let X be a point of the scene, its image X_p can be obtain using the properties of the thin lens, by the intersection of two special rays. The first ray is the ray going through X and parallel to the optical axis, the second is the ray going through X and through the lens centre C (see figure 2.4). With this construction we obtain the *thin lens equation*:

$$\frac{1}{Z} + \frac{1}{Z'} = \frac{1}{f} \tag{2.3}$$

The image of a point in the scene at a distance Z of the centre of the lens will be in sharp focus at a distance Z' (which depends also on the focal length f) of this centre.

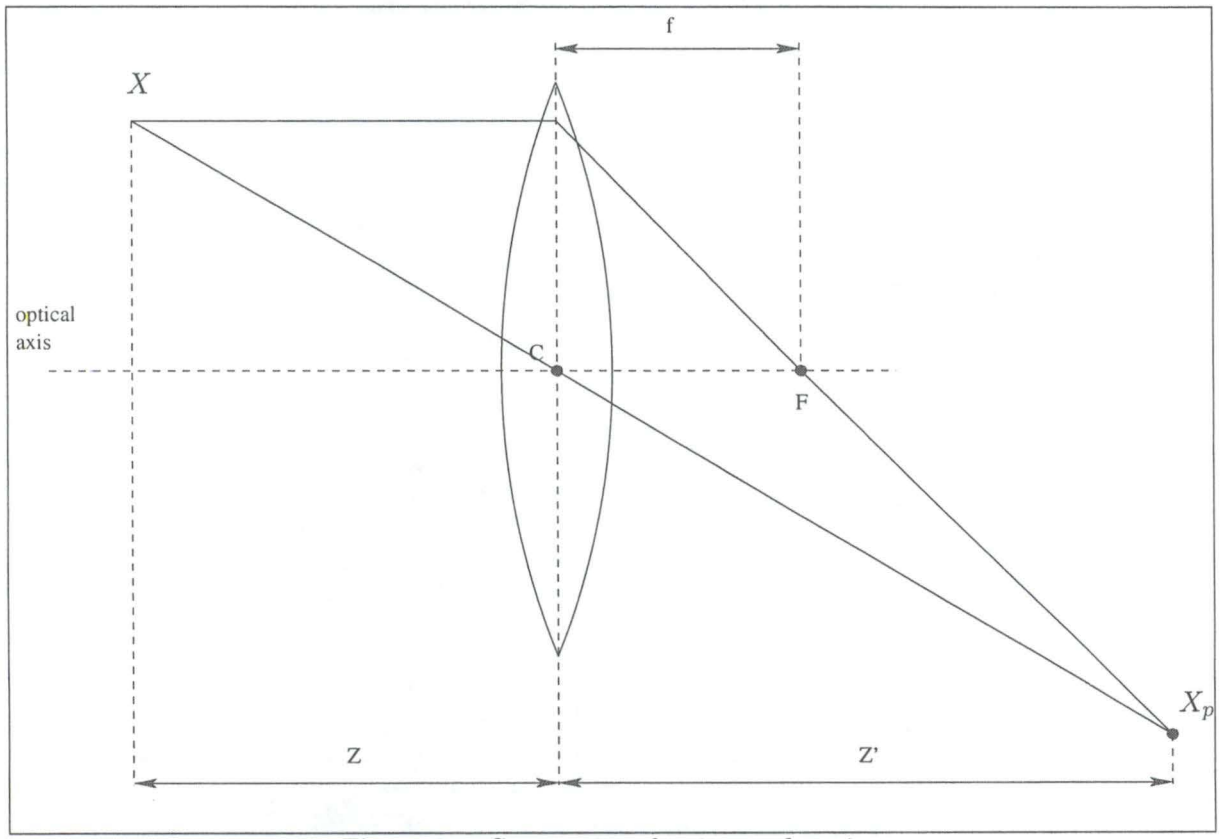


Figure 2.4: Construction of an image of a point.

2.1.3 Digital image

A digital image acquisition system is composed of three hardware devices (see figure 2.5):

- a viewing camera;
- a frame grabber;
- a host computer.

The camera consists of an optical system (approximated by a thin lens) and a CCD (*Charged Coupled Device*) array that constitutes the image plane. The CCD array can be seen as a $(n \times m)$ grid of photosensitive cells. Generally it has a dimension of 1×1 cm and is composed from about 5×10^5 elements.

Each cell receives the light rays and will convert the light energy into voltage, the output of the CCD array is then an analog electrical signal obtained by scanning the lines of photo-sensor and finding the cell's voltage.

The video signal is sent to a device called the *frame grabber*. It will digitised this signal into a rectangular array of $N \times M$ (typically 512×512) integer values and store it in a memory buffer. The elements of this array are called *pixels* (picture elements) and their value

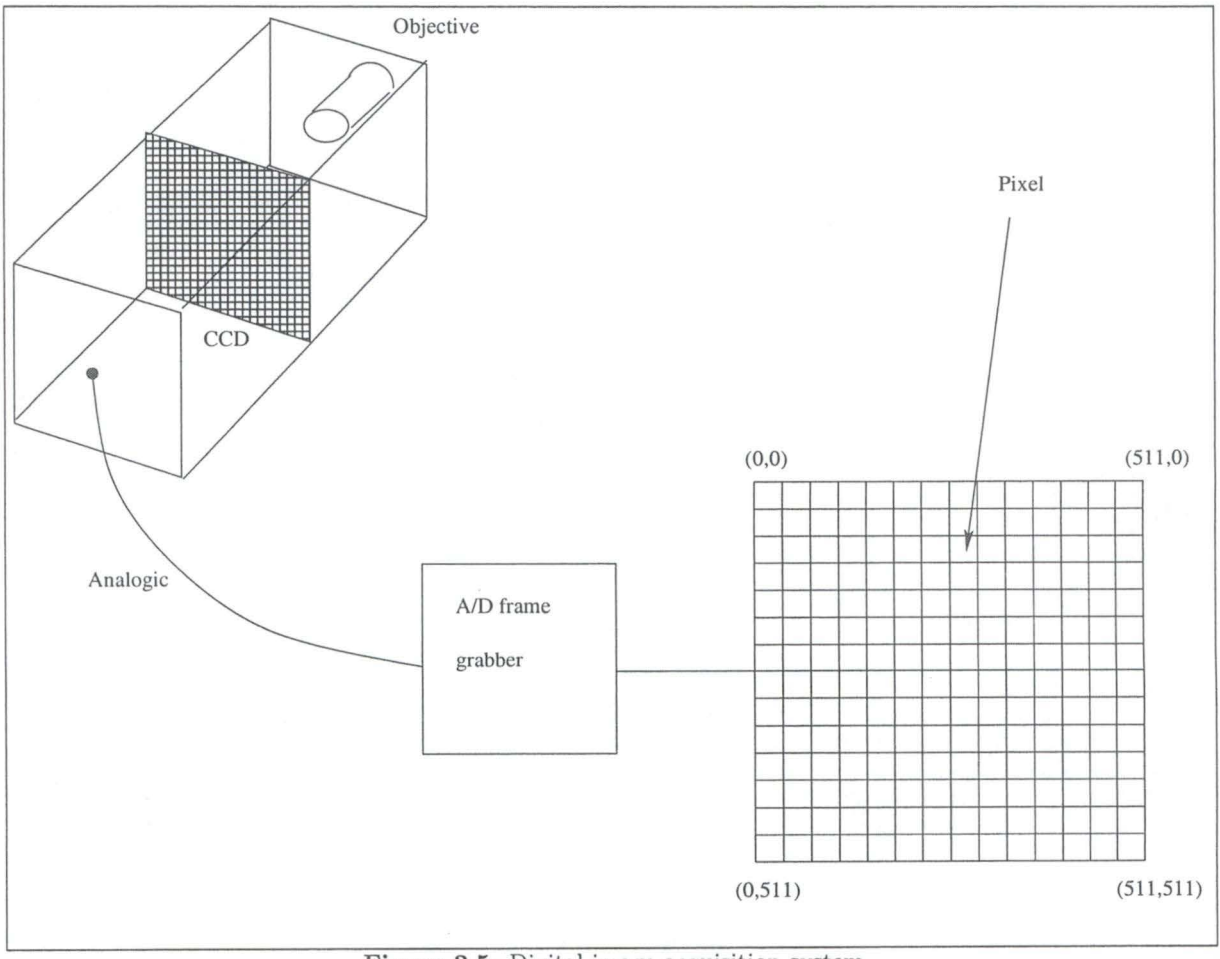


Figure 2.5: Digital image acquisition system

frame buffer.

It's important to note that the dimension of the CCD array ($n \times m$) is not necessary the same than the dimension of the image ($N \times M$), thus the position of a point in the image plane is different if it is measured in CCD elements or in pixels. There is a scale factor between the two measures relied by the following equations

$$\begin{aligned} u_{pix} &= \frac{m}{M} u_{ccd} \\ v_{pix} &= \frac{n}{N} v_{ccd} \end{aligned}$$

We assume that the size of the CCD array element is the *effective pixel size* (measured in $m/pixel$). The process of sampling the image plane and transforming it to the digital format is called the *pixelization.*

2.2 Camera model

between a 3D point and its projection onto the image plane.

As we saw in section 2.1.1, the pinhole camera model can easily be modelled with its optical centre C and its image plane $\mathcal{I}$ (see figure 2.6).

A 3D point X is projected on a point X_p of the image described as the intersection of the line containing C and X with the image plane $\mathcal{I}$.

The intersection between the optical axis and $\mathcal{I}$ is called the *principal point.*

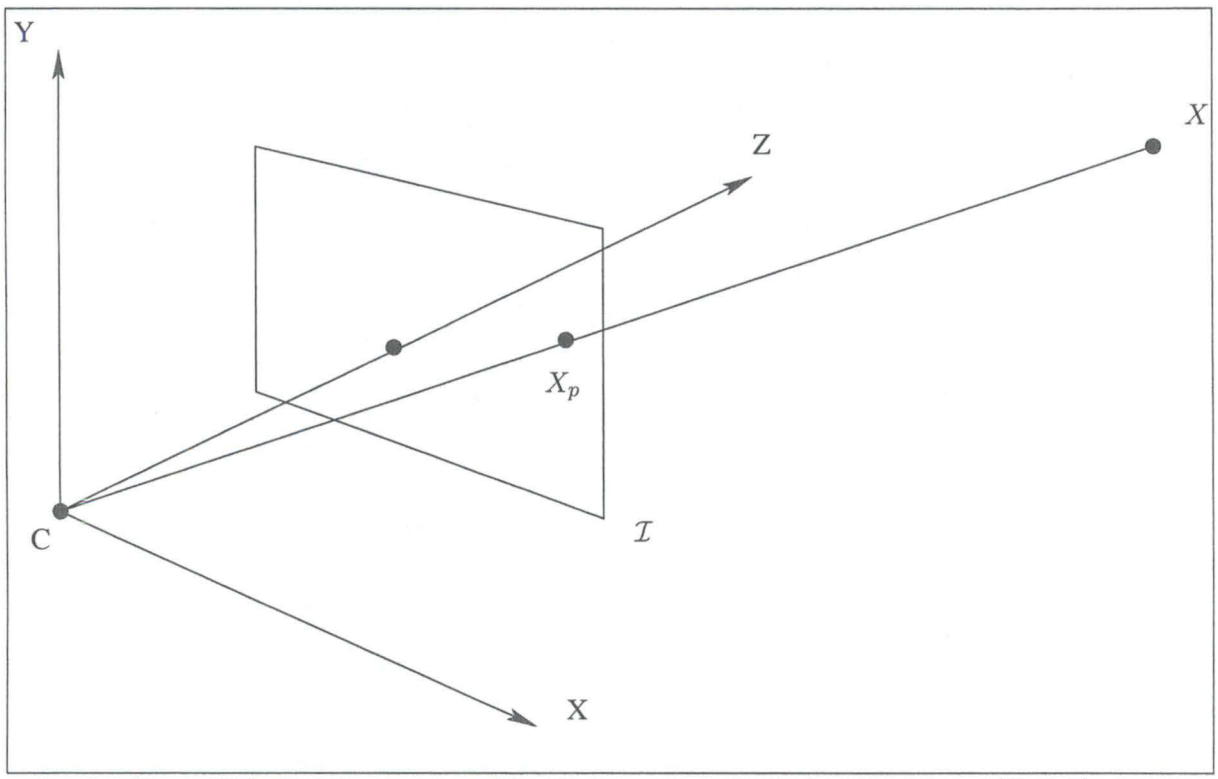


Figure 2.6: Pinhole camera model, with the *camera standard reference frame* (X, Y, Z)

In a camera model, it's important to know the different existing coordinates systems (called reference frames). In this case, three reference frames are used (see figure 2.7):

- **World reference frame** (x, y, z) is an arbitrary 3D reference frame in which a 3D point can be expressed;
- **Image reference frame** (u, v) is the coordinate system in which the position of an image point is expressed;
- **Camera standard reference frame** (X, Y, Z) is a particular 3D reference frame bound

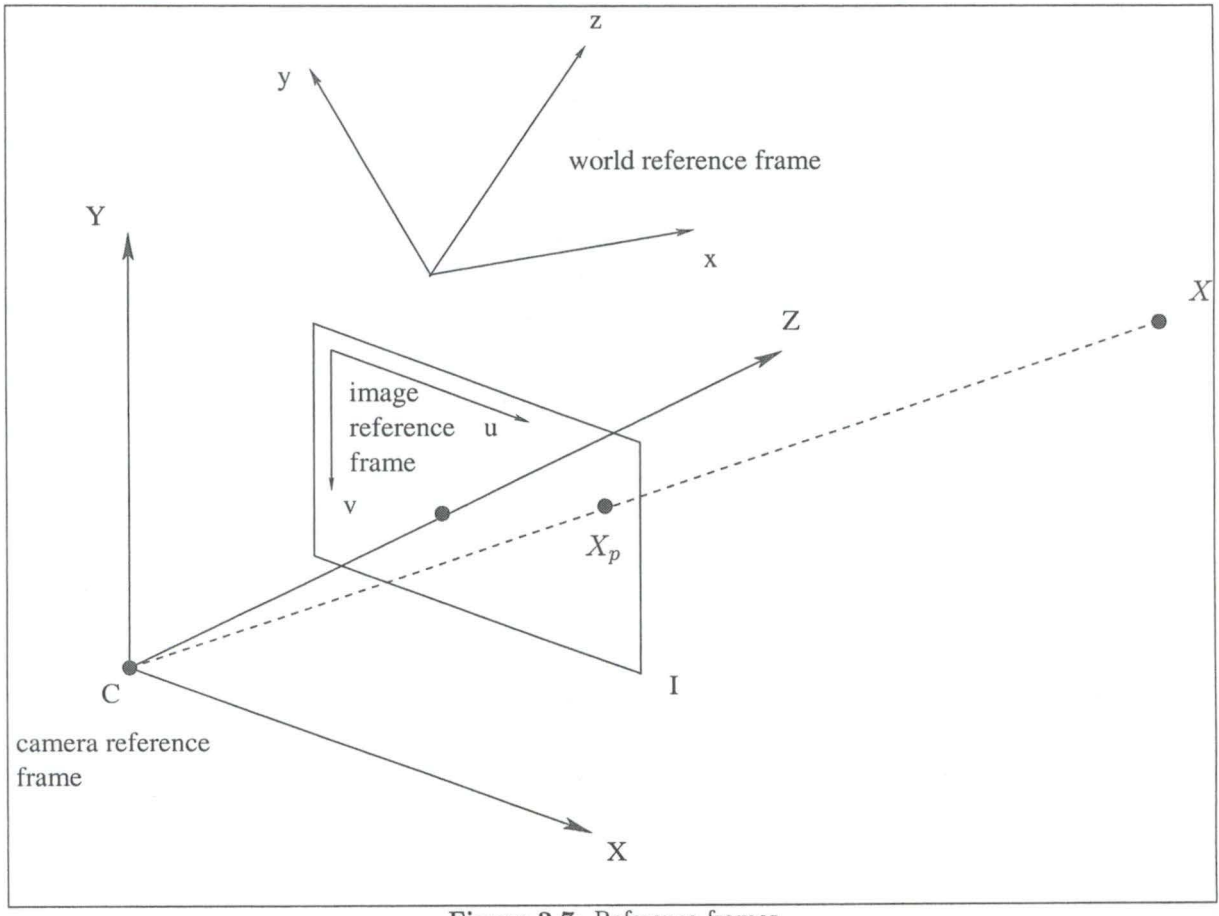


Figure 2.7: Reference frames

First, we will consider a very special case in which the world reference frame is set to the camera standard reference frame, the focal length is equal to 1, the effective pixel size is 1 and the image reference frame is centred in the principal point.

Let $w = (x, y, z)$ be the coordinates of the point X in the world reference frame, and $m = (u, v)$ be the coordinates of the projection X_p of X in the image (in pixels). From section 2.1.1, we can simply deduce the following relationship

$$\frac{1}{z} = \frac{u}{x} = \frac{v}{y} \tag{2.4}$$

that is

$$\begin{aligned} u &= \frac{1}{z}x \\ v &= \frac{1}{z}y \end{aligned} \tag{2.5}$$

Let

$$\tilde{m} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad \text{and} \quad \tilde{w} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.6}$$

be the homogeneous coordinates[1] of X_p and X respectively. We will use $\tilde{}$ to denote homogeneous coordinates. The projection equation, in this simplified case, is

$$\begin{bmatrix} ku \\ kv \\ k \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.7}$$

Note that, in this special case, the value k is equal to the third coordinate of X.

Hence, in homogeneous coordinates, the projection equation is

$$k\tilde{m} = \tilde{P}\tilde{w} \tag{2.8}$$

or,

$$\tilde{m} \simeq \tilde{P}\tilde{w} \tag{2.9}$$

where $\simeq$ means "equal up to an arbitrary scale factor".

The matrix $\tilde{P}$ represents the geometric model of the camera, and is called the *Perspective Projection Matrix* (PPM). In this special case, we have

$$\tilde{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = [I|0] \tag{2.10}$$

2.2.1 Intrinsic parameters

In a more realistic model of camera, the image plane is placed *behind* the projection centre at a certain distance f. Projection equations become

$$\begin{cases} u = \frac{-f}{z}x \\ v = \frac{-f}{z}y \end{cases} \tag{2.11}$$

where f is the focal length in meters.
Moreover, pixelization must be taken into account, by introducing a translation of the principal point and a scaling of u and v axes

$$\begin{cases} u = k_u\frac{-f}{z}x + u_0 \\ v = k_v\frac{-f}{z}y + v_0 \end{cases} \tag{2.12}$$

where (u_0, v_0) are the coordinates of the principal point, k_u (k_v) is the inverse of the effective pixel size along the horizontal (vertical) direction, measured in $pixel/m$.
After these changes, the PPM becomes

$$\tilde{P} = \begin{bmatrix} -fk_u & 0 & u_0 & 0 \\ 0 & -fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = A[I|0] \tag{2.13}$$

where

$$A = \begin{bmatrix} -fk_u & 0 & u_0 \\ 0 & -fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

If the CCD grid is not rectangular, u and v are not orthogonal; if θ is the angle they form, then the matrix A becomes

$$A = \begin{bmatrix} -fk_u & fk_u \cot\theta & u_0 \\ 0 & -fk_v/\sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

Hence, the matrix A has -in general- the following form

$$A = \begin{bmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

where $\alpha_u = -fk_u$, $\alpha_v = -fk_v/\sin\theta$ are the focal lengths in horizontal and vertical pixels, respectively, and $\gamma = fku\cot\theta$ is the *skew factor*. The parameters α_u, α_v, γ, u_0 and v_0 are called the intrinsic parameters.
Note that the CCD grid is often rectangular.

2.2.2 Extrinsic parameters

The previous section was particular since we assumed that the world reference frame coincided with the camera standard reference frame. If we change the camera reference frame to a new world reference frame, the rigid transformation encodes the camera's position and orientation. This transformation is defined by a 3×3 orthonormal rotation matrix $\mathcal{R}$ and a translation vector $\mathcal{T}$.

If w and w' are the Cartesian coordinates of the scene point X in these two frames, we have

$$w = \mathcal{R}w' + \mathcal{T} \tag{2.17}$$

using homogeneous coordinates, the later rewrites

$$\tilde{w} = G\tilde{w}' \tag{2.18}$$

where

The PPM yielded by this reference change is

$$\tilde{P} = A[I|0]G = A[\mathcal{R}|\mathcal{T}] = [A\mathcal{R}|A\mathcal{T}] \tag{2.20}$$

The three entries of the translation vector $\mathcal{T}$ and the three parameters [2] that encode $\mathcal{R}$ are the extrinsic parameters.

Since $\tilde{w}' = G^{-1}\tilde{w}$, with

$$G^{-1} = \begin{bmatrix} \mathcal{R}^T & -\mathcal{R}^T\mathcal{T} \\ 0 & 1 \end{bmatrix} \tag{2.21}$$

the columns of $\mathcal{R}^T$ are the coordinates of the standard reference frame relative to the world reference frame and $-\mathcal{R}^T\mathcal{T}$ is the position of the optical centre C in the world reference frame.

2.3 Epipolar geometry

Let us consider a stereo system composed with two pinhole cameras (see figure 2.8). C and C' are the optical centre of the left and the right camera respectively. A 3D point X is projected onto the two image planes at points x_1 and x_2 which constitute a conjugate pair.

These two perspective images of the rigid scene are related by the so-called *epipolar geometry*, which can be described by a 3×3 singular matrix. It contains all the necessary geometric information to establish correspondence between the two images.

2.3.1 Definitions

- The *epipolar plane* of X is the plane defined by the 3D point X and the optical centres C and C'.
- The *epipole* is the point of intersection of the line joining the two optical centres, that is the *baseline*, with the image planes (see figure 2.8). Thus the epipole is the image, in one camera, of the optical centre of the other camera (note that the epipole may be out of the image plane[3]).

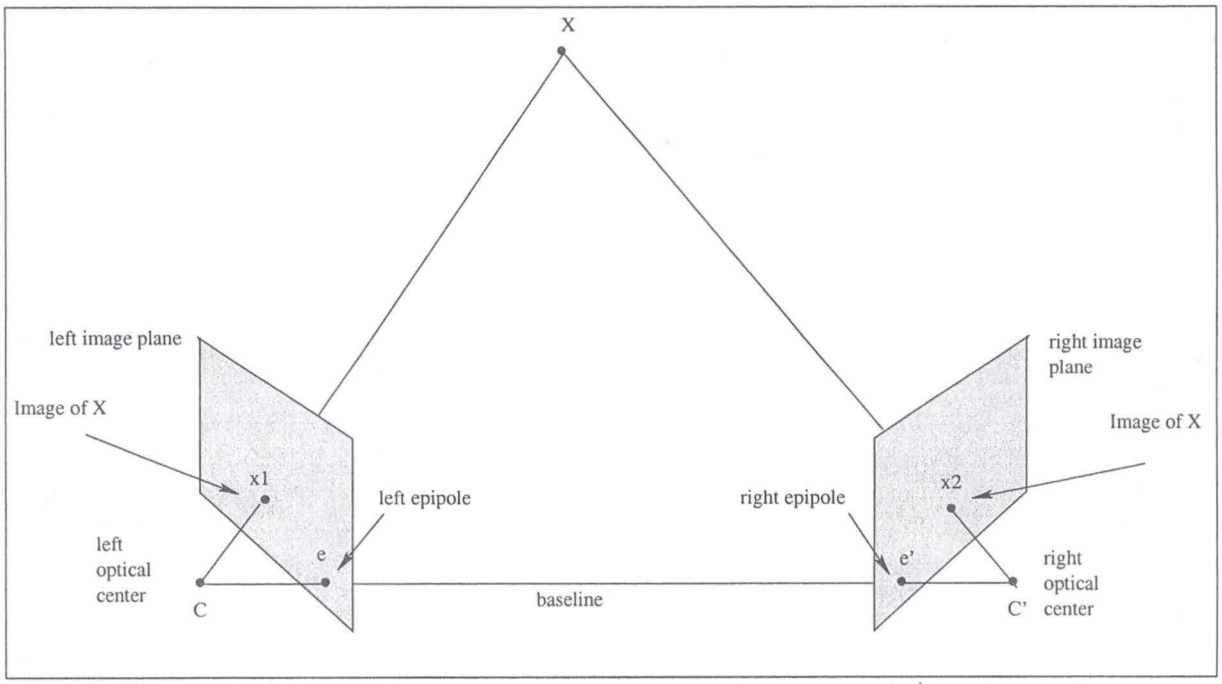


Figure 2.8: The baseline joins the two optical centres. The point e and e' are the epipoles.

- The *epipolar line* of X is the straight line of intersection of the epipolar plane with the image plane (see figure 2.9). It is the image in one camera of a ray through the optical centre and image point in the other camera. All epipolar lines intersect at the epipole.

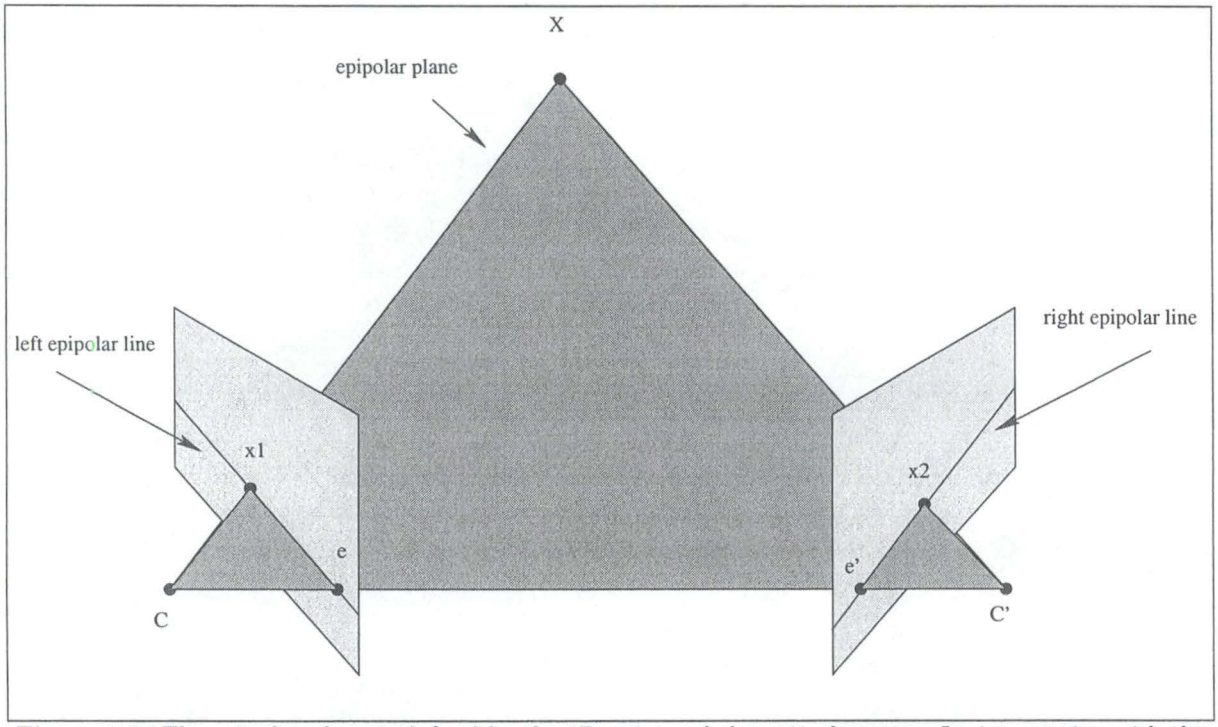
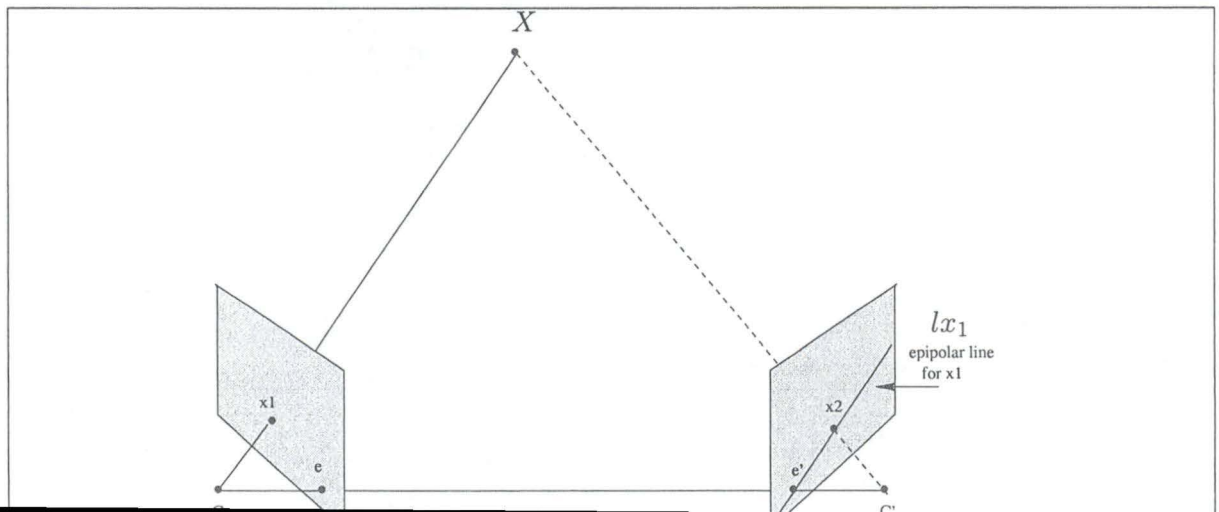


Figure 2.9: The epipolar plane is defined by the 3D point and the optical centres. Its intersection with the image plane constructs the epipolar line.

Thus, an epipolar constraint is that a point x_1 in one image generates a line of possible objects, Cx_1, whose image generates a line lx_1 in the other image on which its corresponding point x_2 must lie (see figure 2.10). We see that this constraint allows us to search the correspondences along a line instead of a region with condition that we know the respective positions of the cameras (*epipolar constraint*).



2.3.2 The epipolar geometry and the fundamental matrix

In this section we will follow [FL95] to define the relation between the two images x_1 and x_2.

Previously, we saw that the epipolar line lx_1 generated by the point x_1 is the projection in the second image of a line Cx_1 through the optical centre C and the point x_1. In other words, lx_1 is the line through the projection of the optical centre C (the epipole e') and the projection of the point of infinity of Cx_1. This consideration will be used later.

First, we have to see how the optical centre can be recovered from the projection matrix $\tilde{P}$ (PPM) defined in section 2.2. In a general way, let us decompose $\tilde{P}$ as follows:

$$\tilde{P} = [Pp] \tag{2.22}$$

where P is a 3×3 matrix of rank 3 and p is a 3×1 vector. Without loss of generality we assume that C is not at infinity, thus we can write

$$\tilde{C} = [C^T 1] \tag{2.23}$$

where $\tilde{C}$ is the projective representation of C according to homogeneous coordinates. Note that C is a 3×1 Euclidean vector of coordinates and the 1 means that C is not at infinity. C satisfies the equation

$$\tilde{P}\tilde{C} = 0 \tag{2.24}$$

from which we conclude

$$C = -P^{-1}p \tag{2.25}$$

Now we come back to the epipolar line. We begin with the computation of e' i.e. the projection of the optical centre C. Using the equations (2.23) and (2.25) and $\tilde{P}'$, the projection matrix of the second image, we write

$$e' = \tilde{P}' \begin{bmatrix} -P^{-1}p \\ 1 \end{bmatrix} = p' - P'P^{-1}p \tag{2.26}$$

where P' and p' come from the application of the equation (2.22) to the PPM $\tilde{P}'$ of the second image.

We also need the projection of the point of infinity of Cx_1. This projection is

$$\tilde{P}' \begin{bmatrix} P^{-1}x_1 \\ 0 \end{bmatrix} = P'P^{-1}x_1 \tag{2.27}$$

We can easily obtain the projective representation of the epipolar line lx_1 which is the cross-product of these two points [4]

$$lx_1 = [p' - P'P^{-1}p] \times P'P^{-1}x_1 = \underbrace{[p' - P'P^{-1}p]_\times P'P^{-1}}_{F} x_1 \tag{2.28}$$

Thus we can define the *Fundamental matrix* F such that

$$lx_1 = Fx_1 \tag{2.29}$$

where F is a 3×3 matrix of rank 2. Therefore, it is singular.

Since the point x_2 corresponding to x_1 belongs to the line lx_1, it follows that

$$x_2^T F x_1 = 0 \tag{2.30}$$

Note that reversing the role of the two images, the fundamental matrix is changed by its transpose. Thus, transposing the equation (2.30), we obtain

$$x_1^T F^T x_2 = 0 \tag{2.31}$$

This shows that just as in the case of one camera where we can relate the optical centre to the perspective projection matrix $\tilde{P}$, in the two cameras case, we can relate the fundamental matrix to the two projection matrices $\tilde{P}$ and $\tilde{P}'$.

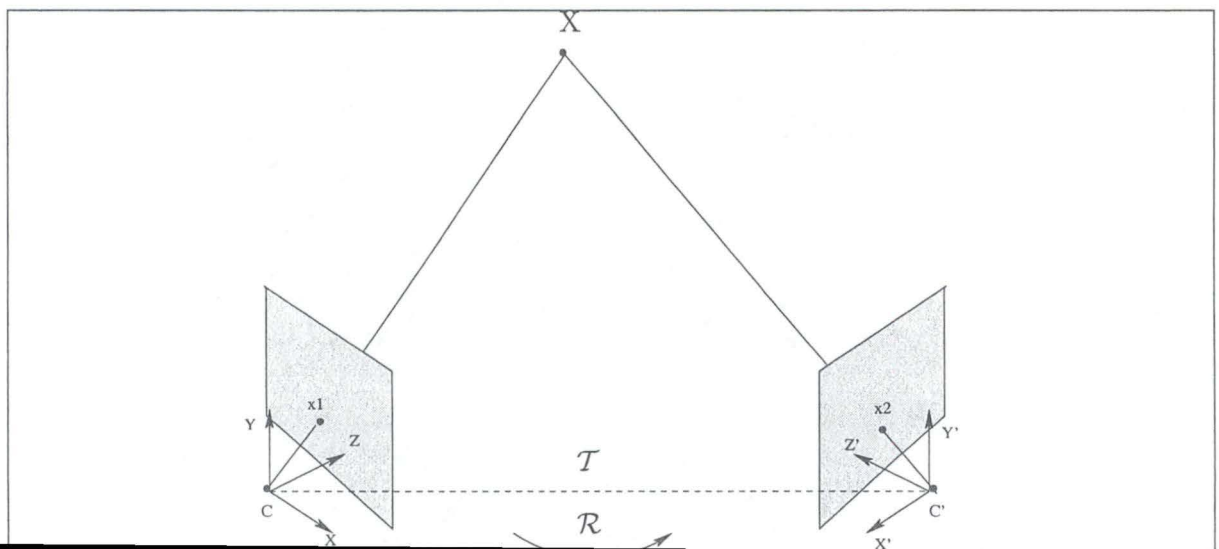
Finally, it's interesting to note that if we work in a calibrated environment (both intrinsic and extrinsic parameters are known), we use the *Essential matrix* E and the equation (2.30) becomes

$$x_2^T E x_1 = 0 \tag{2.32}$$

where

$$E = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \mathcal{R} \tag{2.33}$$

$\mathcal{R}$ and $\mathcal{T} = (t_x, t_y, t_z)^T$ are the orthonormal rotation matrix and the translation vector (see figure 2.11).



2.3.3 Properties

The essential and the fundamental matrices have the following properties:

- The fundamental matrix encapsulates both the intrinsic and the extrinsic parameters of the camera, while the essential matrix encapsulates only the extrinsic parameters;
- The essential matrix E is a 3×3 matrix with only 5 degrees of freedom. To estimate it using the corresponding image points, the intrinsic parameters of both cameras must be known;
- F maps image points to their corresponding epipolar lines, that is, $Fx_1 = lx_1$, since $x_2^T lx_1 = x_2^T Fx_1 = 0$. Likewise, $F^T x_2 = lx_2$ since $l^T x_2 x_1 = 0$;
- F maps epipoles to the origin of the corresponding image plane;
- F has 7 degrees of freedom. There are 9 matrix elements, but only their ratio is significant, which leave 8 degrees of freedom. In addition, noting that $detF = 0$ it remains only 7.

2.4 Image rectification

The idea behind rectification is to define two new perspective matrices which preserve the optical centres and with the baseline contained in the focal planes. This ensures that epipoles are at infinity, hence epipolar lines are parallel. In addition, we choose the convention that the epipolar lines have to be horizontal.

The new perspective projection matrices will have both same orientation but different positions. Positions (optical centres) are the same as the old cameras, but orientation changes because we rotate both cameras around the optical centres in such a way that focal planes become coplanar. In order to simplify the algorithm, the rectified perspective projection matrices will have the same intrinsic parameters. The new camera pair can be thought as a single camera translated along the X axis of its standard reference system (see figure 2.12).

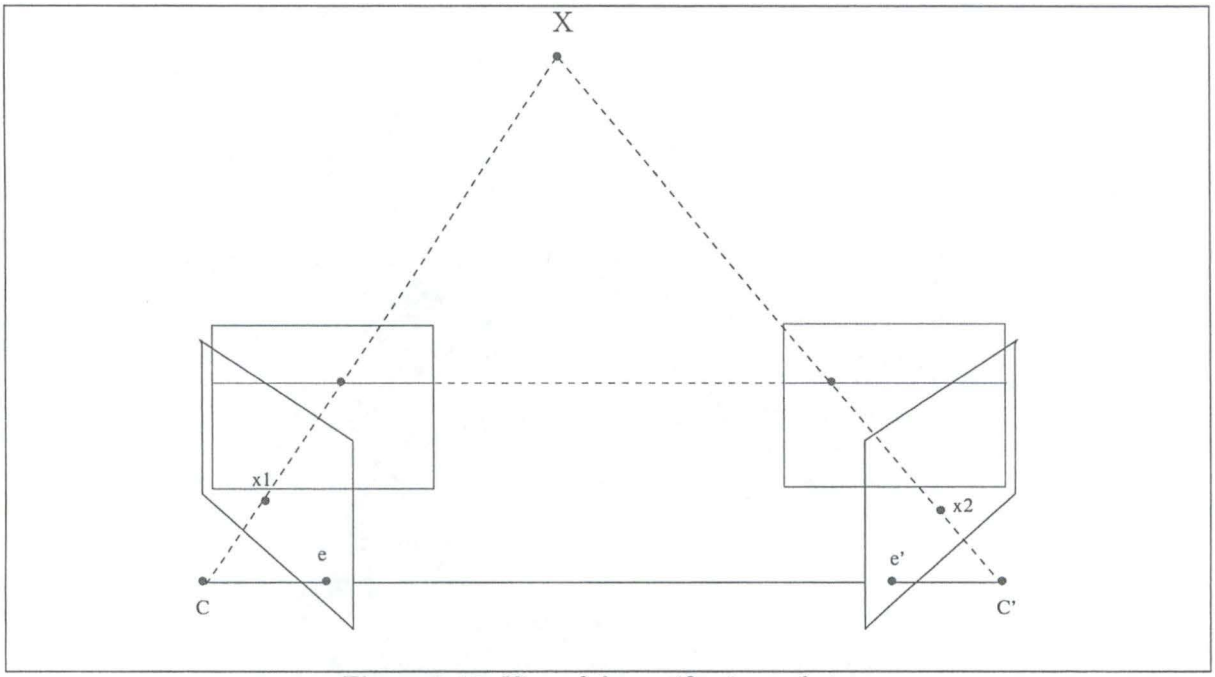


Figure 2.12: View of the rectification technique

Technique

The method consist in the computation of a *rectification matrix* which is used to transform the two images (in fact the perspective projection matrices). The algorithm consists of four steps:

- Rotate the left camera so that the epipole goes to infinity along the horizontal axis;
- Apply the same rotation to the right camera to recover the original geometry;
- Rotate the right camera by $\mathcal{R}$ which is the 3×3 rotation matrix between the cameras;

To carry out this method we construct a triple of mutually orthogonal unit vectors e_1, e_2 and e_3. The first vector is given by the epipole which coincides with the direction of translation:

$$e_1 = \frac{T}{\|T\|} \tag{2.34}$$

The only constraint on the second vector, e_2, is that it must be orthogonal to e_1. So we compute and normalise the cross-product of e_1 with the direction vector of the optical axis:

$$e_2 = \frac{[-T_y, T_x, 0]^T}{\|T\|} \tag{2.35}$$

The third unit vector is determined as:

$$e_3 = e_1 \times e_2 \tag{2.36}$$

Finally the rectification matrix, R_{rect}, is defined as:

$$R_{rect} = \begin{pmatrix} e_1^T \\ e_2^T \\ e_3^T \end{pmatrix} \tag{2.37}$$

This rectification matrix is used to transform the images (with $\mathcal{R}$ which is the rotation matrix between the two cameras):

- Set $R_l = R_{rect}$ and $R_r = \mathcal{R}R_{rect}$;
- For each left-camera point, $p_l = [x, y, f]^T$ compute

  $$R_l p_l = [x', y', z']$$

  and the coordinates of the corresponding rectified point, p'_l, as

  $$p'_l = \frac{f}{z'}[x', y', z'] \tag{2.38}$$

- Repeat the previous step for the right camera using R_r and p_r.

Notice that the rectified coordinates are in general not integer. Therefore, we have to implement *rectification backwards*, that is, starting from the *new* image plane and applying the *inverse* transformations, so that the pixel values in the *new* image plane can be computed as a bilinear interpolation[5] of the pixel values in the *old* image.

Chapter 3

Image registration methods

Introduction

Image registration is an important task when we work with several pictures from the same scene i.e. images which partially or totally overlap. These images can be taken from different viewpoints, different sensors and or at different times. So it finds a variety of applications in computer vision such as image matching for stereovision, pattern recognition, motion analysis, change detection from images taken under different conditions, multi-modal views,...

In our case, when the pictures are taken from different viewpoints, it is useless to simply overlay the images because the result will be full of misregistration. So, the purpose of registration is to find the coordinates transformation that maps each point in one image to a point in the others so as to optimally align the images together when a perfect alignment is not possible.

Following [Bro92], we can distinguish four main classes of registration techniques according to the situation:

multi-modal registration is used to talk about registration of images acquired from different sensors. It integrates information from the different sensors such as positron emission tomography (PET) and single photon emission computed tomography (SPECT) in the medical image analysis to anatomically locate metabolic functions for example;

template registration finds a match for a reference pattern in an image e.g. recognising an airport and its runways in a satellite photograph;

viewpoint registration tackles the matching of several images from different viewpoints. It is well known in computer vision to recover the depth of the scene or to track object motion;

temporal registration handles the case of images taken at different times or under different conditions. For instance, one can register remotely sensed data to monitor evolution e.g. urban growth or pollution.

We can also classify the different registration techniques by taking into consideration the distortions. Even the sensors themselves may introduce noise in the pictures due to the lens and the device in charge of encoding the received signal into a digital picture. We distinguish two types of distortions:

spatial differences which are the cause of misalignment. It is the perspective changes included by the position and the pose of the cameras. This kind of distortions will determine the geometric transformation class such as relative translation, rotation, scale or more complex spatial transformations which will optimally align the images together;

non-spatial differences which are not the source of misregistration but make registration more difficult since the matching will not be exact, even after spatial transformation. It is important to notice that the non-spatial distortions should not be removed by registration. Therefore, the similarity measure and the search space and strategy have

The *compound eye camera* for rescue activities is in the **viewpoint registration** category, with a lot of spatial differences. Subsequently, this review of standard methods is focused on this part. The reader who is interested in the other fields can refer to [Bro92].

The transformations that relate images taken from different viewpoints are local or global geometric transformations accounting for perspective distortions. On the one hand, [Bro92] defines:

global transformation as a single equation which maps the entire images together;

local transformation as a different mapping depending on the spatial location in the images.

On the other hand,we can distinguish global alignment and local alignment:

global alignment establishes a mapping between each image and an arbitrary reference image. In most cases, it reduces the possible accumulated registration errors and it allows a large motion between successive images. This is typically used for independent cameras;

local alignment finds a transformation between pairs of images in a set of several images. This techniques compensates for small amounts of motion parallax. This is typically used for a single camera in motion.

Combining both global and local alignment can improve the quality of the registration. First, global alignment is applied to the whole sequence of images and then local alignment is based on the results of pairwise local image registrations.

In this chapter, we review the most used methods to register images. We begin with the frequency based registration using the Fourier transform. Next we come back in the spatial domain with the presentation of the optical flow and its use in image registration. The two most common methods directly based on intensity are also detailed, namely the correlation-based method and the technique using the *Levenberg-Marquardt* algorithm. Then we describe the feature based registration, especially using corners. All of these methods rely on a score to measure the similarity or the error obtained with the estimated parameters of the registration, so different similarity measures are discussed in the last section.

3.1 Frequency based registration

Many image registration methods work in the spatial domain, we will review them in the next sections. The phase correlation however is based on the Fourier theory and thus works in the frequency domain. The Fourier transform is used to decompose an image into its sine and cosine components and has several properties that can be exploited for image registration. This way of processing images is not really intuitive so we will first describe the basic principles and then present the use of the Fourier transform in image registration.

3.1.1 The Optical Fourier Transform

A simple lens can instantaneously achieve a Fourier Transform. The input image should be placed at the focal length of the lens and illuminated with a coherent light incident parallel to the optical axis. The corresponding Fourier image can be captured on a frosted glass screen placed at the opposite focus of the lens (see figure 3.1 and figure 3.2). Here is the intuitive explanation given by S. Lehar in [Leh]: «*Every point on the input image radiates an expanding cone of rays towards the lens, but since the image is at the focus of the lens, those rays will be refracted into a parallel beam that illuminates the entire image at the ground glass screen*».

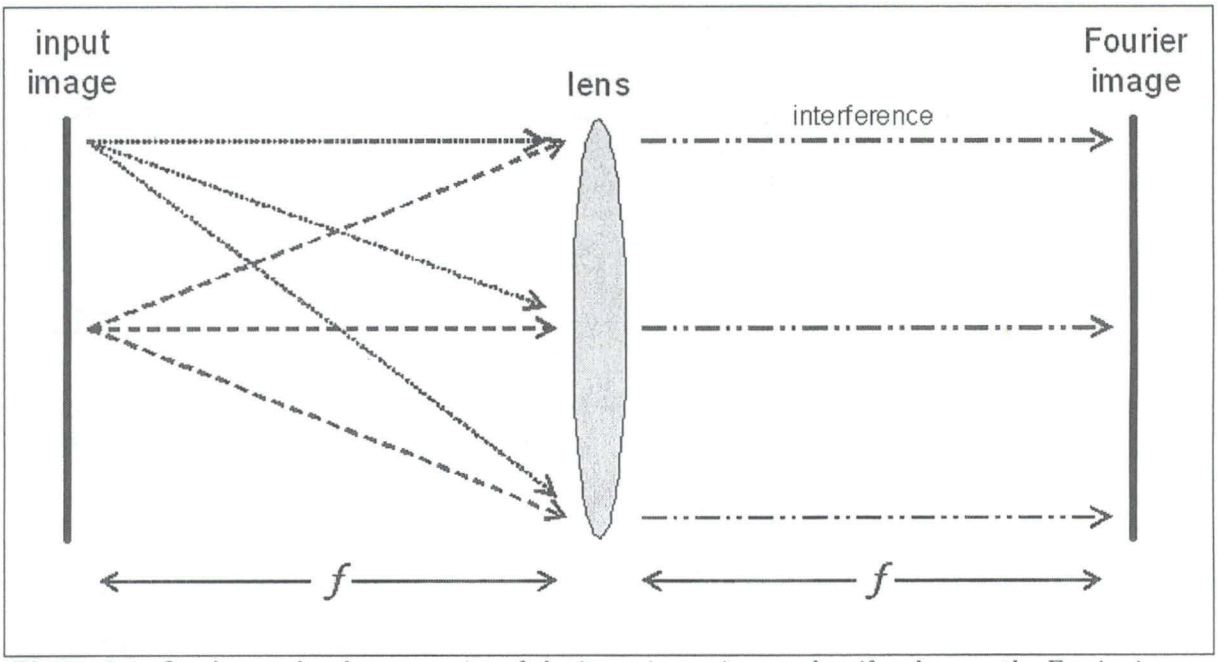


Figure 3.1: On the one hand, every point of the input image is spread uniformly over the Fourier image, where constructive and destructive interference will automatically produce the proper Fourier representation.

The forward spatial 2D Fourier transform $\Im$ of $f(x, y)$ is defined as:

$$\Im\{f(x,y)\} \triangleq F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-i2\pi(ux+vy)}dxdy \tag{3.1}$$

where x, y are the spatial variables and u, v are the spatial frequencies (radians per unit

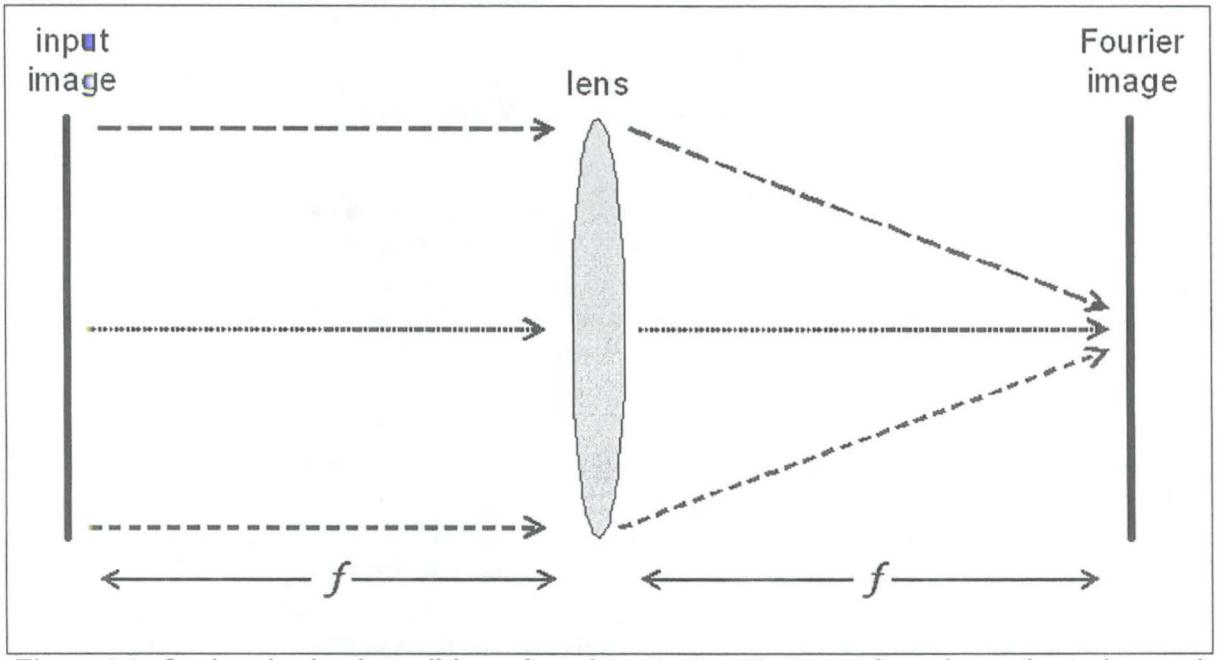


Figure 3.2: On the other hand, parallel rays from the entire input image are focused onto the single central point of the Fourier image, where it defines the central DC component (the DC component is the name given for the $F(0,0)$ value which is the sum of all frequencies) which is the average brightness of the input image.

This optical Fourier transform can automatically perform the inverse Fourier transform if we operate in the reverse direction, recovering the brightness image from the Fourier representation. The forward and inverse Fourier transforms are identical except for a minus sign that reverses the direction of the computation. That is, the inverse Fourier transform is given by the formula:

$$\Im^{-1}\{F(u,v)\} = f(x,y) = \frac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u,v) e^{i2\pi(ux+vy)} du dv \tag{3.2}$$

3.1.2 Basic principles

According to the Fourier theory, any signal can be expressed as a sum of a series of sinuscids. In 2D, this periodic function is of the form [HE]:

$$s(x,y) = A\cos(2\pi f_x x + 2\pi f_y y + \theta) \tag{3.3}$$

where A is the amplitude, f_x and f_y are the horizontal and vertical frequencies respectively and θ is the phase. The periods over the horizontal and vertical axis of the sinusoid are defined using the reciprocal of frequency i.e. $\frac{1}{f_x}$ and $\frac{1}{f_y}$. Since we are working with digital images, we are only interested in discrete sinusoids and we can write the frequencies f_x and f_y as $\frac{u}{N}$ and $\frac{v}{M}$ where N, M are the width and height of the image and u, v are integers that index vertical and horizontal frequencies. Let the input image $I(x,y)$ with $0 \leq x < N$, $0 \leq y < M$

with $\theta = 0$ or $\frac{\pi}{2}$ as follows:

$$I(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \left(a(u,v) \cos\left(2\pi \frac{ux}{N} + 2\pi \frac{vy}{M}\right) + b(u,v) \sin\left(2\pi \frac{ux}{N} + 2\pi \frac{vy}{M}\right)\right) \tag{3.4}$$

where $a(u,v)$ and $b(u,v)$ are the weights of the sinusoids and their absolute values indicate the amount of each frequency in the image $I(x,y)$. A high frequency image has large weights for high frequency indices u, v and similarly a low frequency image has large weights for small frequency indices u, v. Note that the weights are defined on a coordinates system indexed by pairs (u,v) which is called the frequency domain.

The *Discrete Fourier Transform* (DFT) is the operation of obtaining the frequency domain representation of the image. The following definition is from the HIPR[1] website [FPWW02]: «*The DFT is the sampled Fourier transform and therefore does not contain all frequencies but only a set of samples which is large enough to fully describe the real domain image. The number of frequencies corresponds to the number of pixels in the real domain image*». Consequently, the brightness image and the Fourier image are of the same size and both contain exactly the same information, but expressed in terms of amplitude and phase. In other words, the Fourier image is a function of spatial frequency (i.e. a function from the frequency plane to complex numbers), rather than brightness as a function of spatial displacement (i.e. a function from the 2D plane to real numbers). Therefore, an inverse Fourier transform (3.6) of the image in the frequency space exists and produces an exact pixel-for-pixel copy of the original image.

For a $N \times M$ image $I(x,y)$, the two dimensional $N \times M$ DFT $F(u,v)$ is given by:

$$F(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} I(x,y) e^{-i2\pi(\frac{ux}{N} + \frac{vy}{M})} \qquad \text{for } u = 0, \ldots, N-1 \text{ and } v = 0, \ldots, M-1 \tag{3.5}$$

where $I(x,y)$ is the image in the real space and the exponential term is the basis function corresponding to each point $F(u,v)$ in the frequency space[2]. This equation can be interpreted as: «*the value for each point $F(u,v)$ is obtained by multiplying the real image with the corresponding base function and summing the result*». This is an operation called "*convolution*".

Each term of the Fourier transform depends on

- the *spatial frequency*. It corresponds to the frequency across space with which the brightness modulates i.e. the rate of change of pixel values in the image. For example, rapid changes of gray-level values imply high frequency and almost uniform gray-level values imply low frequency;

and encodes

- the *magnitude* of the sinusoid which stands for the contrast, i.e. the difference between the darkest and the brightest peaks of the image (see equation 3.7);

- the *phase* which represents how the wave is shifted relative to the origin (see equation 3.8);

and the whole series of sinusoids ranges through spatial frequency from zero, this is the DC component $F(0,0)$[3], all way up to the *Nyquist* frequency $F(N-1, M-1)$[4].

Finally, to recover the image in the real space, we can use the inverse DFT as follows:

$$I(x,y) = \frac{1}{NM} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F(u,v) e^{i2\pi(\frac{ux}{N} + \frac{vy}{M})} \qquad \text{for } x = 0, \ldots, N-1 \text{ and } y = 0, \ldots, M-1 \tag{3.6}$$

We can decompose the DFT expression in two parts [Der98, HE]. The magnitude of the complex DFT coefficients is called the *amplitude spectrum* which is defined to be

$$\|F(u,v)\| = \sqrt{Re\{F(u,v)\}^2 + Im\{F(u,v)\}^2} \tag{3.7}$$

The square of the amplitude spectrum is called the *power spectrum* or the *spectral density* of the image. The phase of the DFT coefficients, namely the *phase spectrum* of the image, corresponds to

$$\angle F(u,v) = \tan^{-1}\left(\frac{Im\{F(u,v)\}^2}{Re\{F(u,v)\}^2}\right) \tag{3.8}$$

In addition, the DFT and inverse DFT are periodic. A conjugate symmetry relation between the DFT coefficients exists over one period of the DFT. This relation is formally stated as:

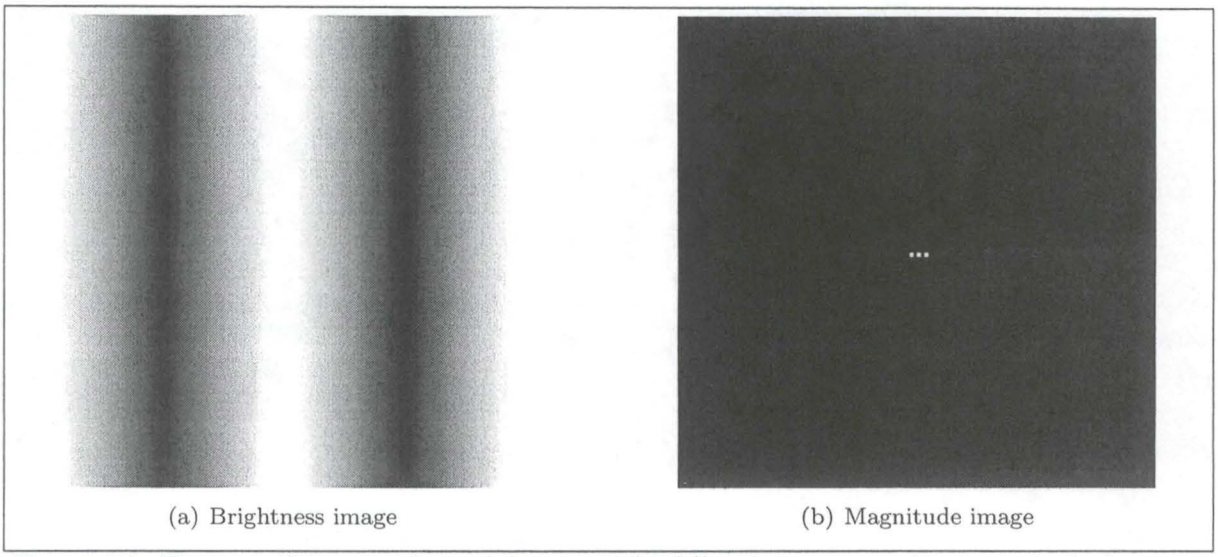
$$F(u,v) = F^*(N-u, M-v) \qquad \text{for } u \in [0,N] \text{ and } v \in [0,M] \tag{3.9}$$

where * denotes complex conjugate. This conjugate symmetry implies:

$$\|F(u,v)\| = \|F(N-u, M-v)\| \tag{3.10}$$

On figure 3.3, we can see a simple example of the Fourier transform found in [Leh]. Let a sinusoidal brightness image of spatial frequency 1 be the image in the real space and its corresponding image in the frequency space.

[3]No modulation i.e. the average brightness of the whole image. For example, a zero DC component would



(a) Brightness image (b) Magnitude image

Figure 3.3: The sinusoidal image of spatial frequency 1 (3.3(a)) and its corresponding image in the frequency space (3.3(b)). Every pixel in the Fourier image is a spatial frequency value, the magnitude of that value is encoded by the brightness of the pixel (with white color for high value). The bright pixel at the very center is the DC component. The two bright pixels either side of the center encode the sinusoidal pattern. The brighter the peaks in the Fourier image, the higher the contrast in the brightness image. All other pixels in the Fourier image are black i.e. values are zero because there is only one Fourier component in this simple example.

A signal containing only a single spatial frequency of frequency f is plotted as a single peak at point f along the spatial frequency axis, the height of that peak corresponding to the amplitude, or contrast of that sinusoidal signal. Actually, due to the conjugate symmetry property, the Fourier transform also plots a mirror-image of the spatial frequency plot reflected across the origin, with spatial frequency increasing in both directions from the origin. These two plots are always mirror-image reflections of each other, with identical peaks at f and $-f$ as shown on figure 3.4.

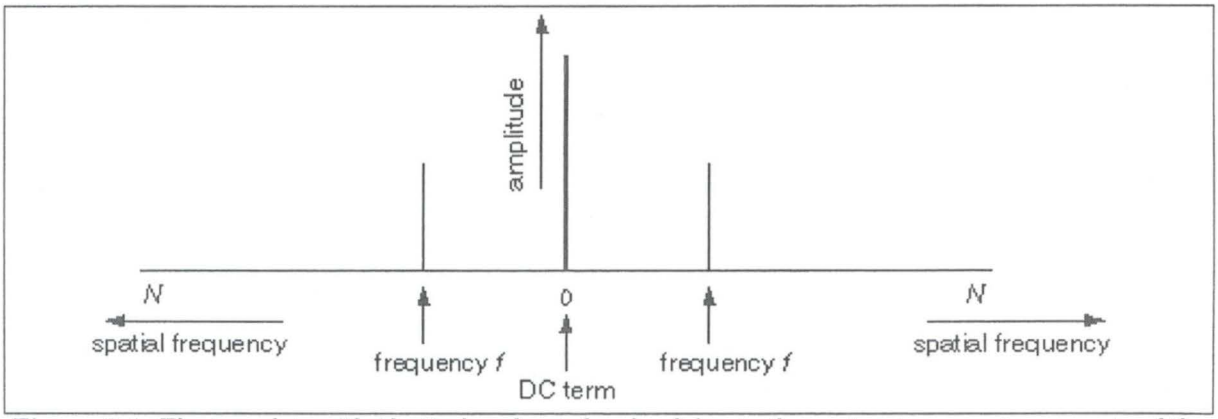
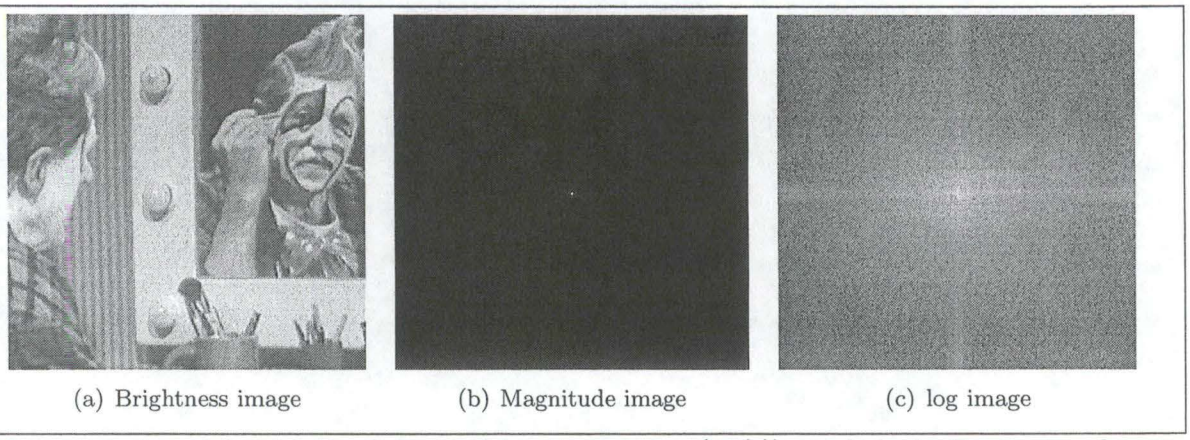


Figure 3.4: The two plots with identical peaks at f and $-f$ due to the conjugate symmetry property of the Fourier transform.

Note that a logarithmic transformation is commonly applied to the Fourier image because

example found in [FPWW02].



(a) Brightness image (b) Magnitude image (c) log image

Figure 3.5: From left to right: the original brightness image (3.5(a)) to which the Fourier transform is applied. The magnitude image (3.5(b)) calculated from the complex result. We can only see the value of the DC component which is by far the largest component of the image. A logarithmic transformation is applied to obtain the last image (3.5(c)). We can see that the image contains components of all frequencies, but that their magnitude gets smaller for higher frequencies. Hence, low frequencies contain more image information than higher ones. We can also observe that there are two dominating directions originating from the regular pattern in the background of the original image.

One of the interesting properties of the DFT is to be separable [FPWW02, HE], i.e. it can be seen as a 1D transform on every row of the image, and another 1D transform on every column of the image, resulting in a 2D Fourier transform of the same size as the original image. Indeed, the equation can be written as:

$$G(u,y) = \sum_{x=0}^{N-1} I(x,y) e^{-i2\pi(\frac{ux}{N})} \qquad \text{for } u = 0, \ldots, N-1 \tag{3.11}$$

$$F(u,v) = \sum_{y=0}^{M-1} G(u,y) e^{-i2\pi(\frac{vy}{M})} \qquad \text{for } v = 0, \ldots, M-1 \tag{3.12}$$

Expressing the 2D Fourier transform in terms of a series of 1D transforms decreases the number of required computations. Even with these computational savings, the 1D DFT has n^2 complexity. This can be reduced to $n \log_2 n$ by using the *Fast Fourier Transform* (FFT) to compute the 1D DFTs. The FFT takes the "divide and conquer" approach to solve the problem. There also exists other common variants such as the *Discrete Cosine Transform* (DCT). It is not the purpose of this part to talk about that, the interested reader can refer to [Der98, FPWW02, HE].

3.1.3 The phase correlation

Now that the principles of the Fourier transform have been presented, we can describe its use in the image registration field. As L. G. Brown recalled it in [Bro92], translation, rotation

Let $I_1(x, y)$ and $I_2(x, y)$ be the intensity functions of two overlapping images, the purpose is to recover the displacement (Δ_x, Δ_y) such that:

$$I_1(x - \Delta_x, y - \Delta_y) = I_2(x, y) \tag{3.13}$$

The phase correlation algorithm can estimate the translation between a pair of overlapping images. This translation property of the Fourier transform is also called the Shift theorem (3.1). Denote by $\Im\{I(x, y)\}$ the Fourier transform of $I(x, y)$:

$$\Im\{I(x, y)\} \triangleq F(u, v) \tag{3.14}$$

Theorem 3.1 (The Shift theorem) *Let $I(x, y)$ be a function on $\Re^2$ and $F(u, v)$ its 2D Fourier transform, then*

$$\Im\{I(x - \Delta_x, y - \Delta_y)\} = F(u, v)e^{-i2\pi(u\Delta_x + v\Delta_y)} \tag{3.15}$$

In the case of a simple displacement, the Fourier magnitude of the two images in the frequency space is the same. The translation is expressed in the difference between the phases of the images. Let F_1 and F_2 be the DFT of I_1 and I_2 respectively, we have according to the Shift theorem:

$$F_2(u, v) = F_1(u, v)e^{-i2\pi(u\Delta_x + v\Delta_y)} \tag{3.16}$$

According to the Correlation theorem[5], the Fourier transform of two images correlation is [Lia00]:

$$Cross(u, v) = F_1(u, v)F_2^*(u, v) \tag{3.17}$$

where $*$ denotes the complex conjugate. As stated before, this is the phase change in the spectrum domain that reflects the shift in the spatial domain. Thus, the cross-power spectrum (3.17) is normalised by its magnitude to obtain its phase:

$$\Phi[Cross(u, v)] = \frac{F_2(u, v)F_1^*(u, v)}{\|F_2(u, v)F_1^*(u, v)\|} \tag{3.18}$$

Equations (3.16) and (3.18) yield to

$$\Phi[Cross(u, v)] = e^{-i2\pi(u\Delta_x + v\Delta_v)} \tag{3.19}$$

The inverse Fourier transform (IDFT) is applied to the normalised cross-power spectrum to obtain the phase correlation surface:

$$Corr(x, y) = \Im^{-1}\{\Phi[Cross(u, v)]\} \triangleq \Im^{-1}\{e^{-i2\pi(u\Delta_x + v\Delta_y)}\} = \delta(x - \Delta_x, y - \Delta_y) \tag{3.20}$$

Thus applying the inverse Fourier transform to the phase shift results in a δ function offset by exactly the amount of translational displacement. This correlation surface $Corr(x, y)$ is 0 everywhere except a pulse which is located at (Δ_x, Δ_y). Thus we are able to recover the displacement between the two images by seeking the pulse with the equation (3.20).

The peak value of the correlation surface $Corr(m, n)$ should ideally be equal to 1 but the presence of random noise, dissimilar parts and non-translational motion imply a peak value

smaller than 1. Therefore, to work even in the presence of noise, the height of the IDFT values is used as a measure for the quality of the match. So the translation parameters Δ_x and Δ_y are estimated by searching the maximum (see figure 3.6) as follows:

$$(x, y) = \arg\left(\max_{(\tilde{x},\tilde{y})}\left(Corr(\tilde{x}, \tilde{y})\right)\right) \tag{3.21}$$

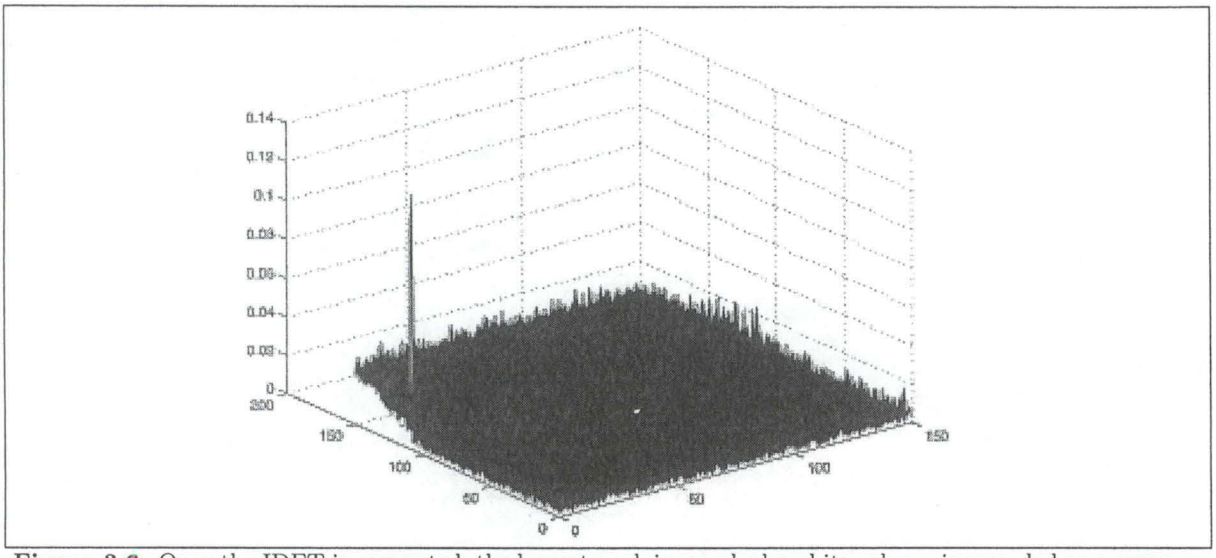


Figure 3.6: Once the IDFT is computed, the largest peak is searched and its value μ is recorded as a measure of the match quality.

In [McG98, RC96], the phase correlation method has been extended to recover rotation, scale and translation parameters between two images, assuming $I_2(x, y)$ be the scaled, rotated and translated replica of $I_1(x, y)$. In a few words, the rotation movement is represented as a translational displacement by converting to polar coordinates. Then it is deduced in a similar manner as translation using phase correlation. It should be possible to estimate any transformation between images as proposed in several works. Though the Fourier approach is suited to estimate general small motion between images (e.g. between frames in a video sequence), it is difficult to extend the method to large general transformations i.e. images taken from very different viewpoints.

3.1.4 Advantages and drawbacks

In short, the frequency domain method or phase correlation estimates the 2D Fourier transform of each image and computes the phase difference at each frequency, performing an inverse Fourier transform and searching for a peak in the magnitude image.

This method is often used to get good initial guesses for pairs of images which overlap by as little as fifty percent. It is particularly well-suited to images taken under different conditions of illumination or images taken with different sensors since the phase difference for every frequency contributes equally and it is insensitive to changes in spectral energy.

As we have seen in this section, the phase correlation is usually limited to translation wh it is possible to estimate scale changes and rotations through the conversion of the Cartesi to polar coordinates system. Thus, it is a robust method in the case of simple transformatio such as translation and rotation between a pair of images (i.e. Euclidean transformatio but it is more complicated to tackle problems such as the projective case. If the inter-fra motion is not mostly translational (i.e. large rotations or zooms), the use of the correlati measure is not possible.

3.2 Optical flow based registration

Here, the image registration is seen from the motion point of view. The optical flow recovery consists in estimating a general motion between two images and generally involves estimating an independent displacement vector for each pixel. Most of them are gradient-based algorithm which estimates the optical flow from spatial (and temporal) derivatives of the image intensity. In this section we will first review the underlying theory according to [ZB01] and then relate the gradient-based method in detail.

3.2.1 The optical flow

Since the coordinates system is centred on the viewer i.e. the camera, a 3D point X with homogeneous[6] position vector $\vec{X} = (X, Y, Z, W)^T$ moves along a 3D path when the camera moves. Note that W is assumed to be constant i.e. W is set to 1. The derivative of this path with respect to time corresponds to the instantaneous 3D velocity of the point.

$$\vec{\varpi} = \frac{d\vec{X}}{dt} = \left(\frac{dX}{dt}, \frac{dY}{dt}, \frac{dZ}{dt}, 0\right)^T \tag{3.22}$$

Working back in the image coordinates system, the 3D point X is projected through the optical centre here taken as origin onto a 2D projection plane (the image) at distance f, the focal length, along the z axis as follows[7]:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \tilde{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.23}$$

Thus, the 2D point Cartesian coordinates (x, y) under perspective projection are

$$\begin{aligned} x &= \frac{x'}{z'} = f\frac{X}{Z} \\ y &= \frac{y'}{z'} = f\frac{Y}{Z} \end{aligned} \tag{3.24}$$

And we want to recover the velocity vector for every point:

$$\vec{\vartheta} = (\vartheta_x, \vartheta_y)^T \tag{3.25}$$

which represents how quickly the point is moving across the images and the direction of its move.

Let the image intensity $I(x, y, t)$ be now function of time t as well as of x and y. So, for two images I_1 and I_2 taken at time t and $t + dt$ respectively, we have:

$$\begin{aligned} I_1(x, y) &\triangleq I(x, y, t) \\ I_2(x, y) &\triangleq I(x, y, t + dt) \end{aligned} \tag{3.26}$$

A first assumption to derive the relationship between intensity change and optical flow that the time-varying image intensity can be expanded in a first-order Taylor series expansi (3.27):

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \dots \quad (3.2$$

which expresses the intensity of a point a small distance away and a small time later. T dots stand for higher order terms.

The point at a position (x, y) in the image at time t has moved through a distance $(dx, d$ during dt. Moreover, we can suppose that the point intensity is the same at time t and time $t + dt$. This second assumption implies that:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) \quad (3.2$$

Thus we have:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \dots = 0 \quad (3.2$$

The time arguments t and $t + dt$ in the intensity function are actually just indices for t two images (see equations 3.26). Consequently, we can assume a constant velocity for ima points between the two images and the higher order terms are 0. Dividing equation (3.29) dt yields the *optical flow constraint equation*:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \quad (3.3$$

where:

- $\frac{dx}{dt} = \vartheta_x$ and $\frac{dy}{dt} = \vartheta_y$ are the x and y components of the optical flow vector;
- $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the spatial and temporal partial derivatives of image intensity[8] a represent how quickly the intensity changes with space and time.

Thus, as the 3D point moves about images according to their respective coordinates syste its corresponding 2D image point also traces out a 2D path and the vector of 2D velocity defined by the derivative of that path.

$$\vec{\vartheta} = (\vartheta_x, \vartheta_y)^T = \left(\frac{dx}{dt}, \frac{dy}{dt}\right)^T \quad (3.3$$

This is what it is commonly called the optical flow field.

3.2.2 The gradient-based method

Still according to [ZB01], we assume that the only difference between the two images i translation in coordinates. In this way, we can easily explain the estimation of the veloc through gradient-based measurement.

[8] Spatial and temporal partial derivatives can be approximated by differences in practice because ima

We have obtained the optical flow constraint (3.30) also called the *gradient constraint* which is independent of dt and can be write as follows to highlight the velocity components ϑ_x and ϑ_y:

$$\vartheta_x I_x(x,y,t) + \vartheta_y I_y(x,y,t) + I_t(x,y,t) = 0 \tag{3.32}$$

where, for the ease of notation, I_x, I_y and I_t are the spatial and temporal derivatives which are usually noted $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ respectively.

The gradient constraint relates the velocity at one pixel to the spatial and temporal derivatives of image intensity at the same location. Using the gradient constraint at only one pixel to recover the velocity is impossible due to the number of unknowns (ϑ_x and ϑ_y) which is higher than the number of equations. Assuming that nearby points in the image move in a similar manner[9], the solution is to solve velocity over a group of pixels and thus combining information over a spatial region. For example, we can take two neighbouring pixels with respective coordinates (x_1, y_1) and (x_2, y_2) to obtain

$$\begin{pmatrix} I_x(x_1,y_1,t) & I_y(x_1,y_1,t) \\ I_x(x_2,y_2,t) & I_y(x_2,y_2,t) \end{pmatrix} \begin{pmatrix} \vartheta_x \\ \vartheta_y \end{pmatrix} + \begin{pmatrix} I_t(x_1,y_1,t) \\ I_t(x_2,y_2,t) \end{pmatrix} = \vec{0} \tag{3.33}$$

An example of the optical flow pattern from the sequence of images of a rotating Rubik's cube shown in figure 3.7 is given in figure 3.8.

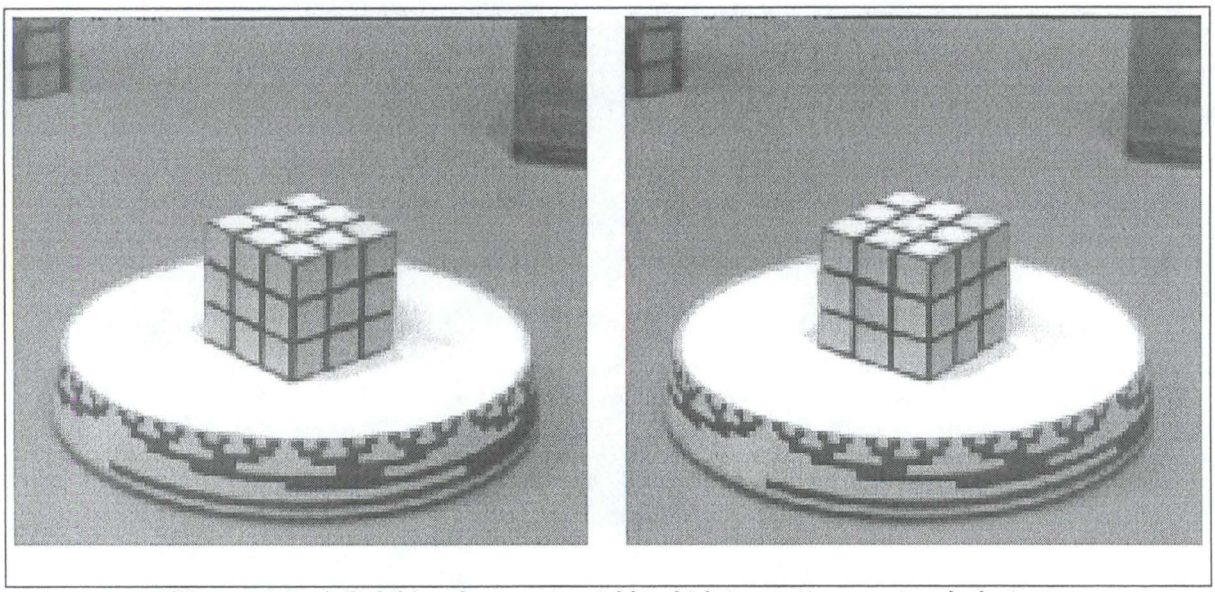


Figure 3.7: A Rubik's cube on a turntable which is rotating counter-clockwise.

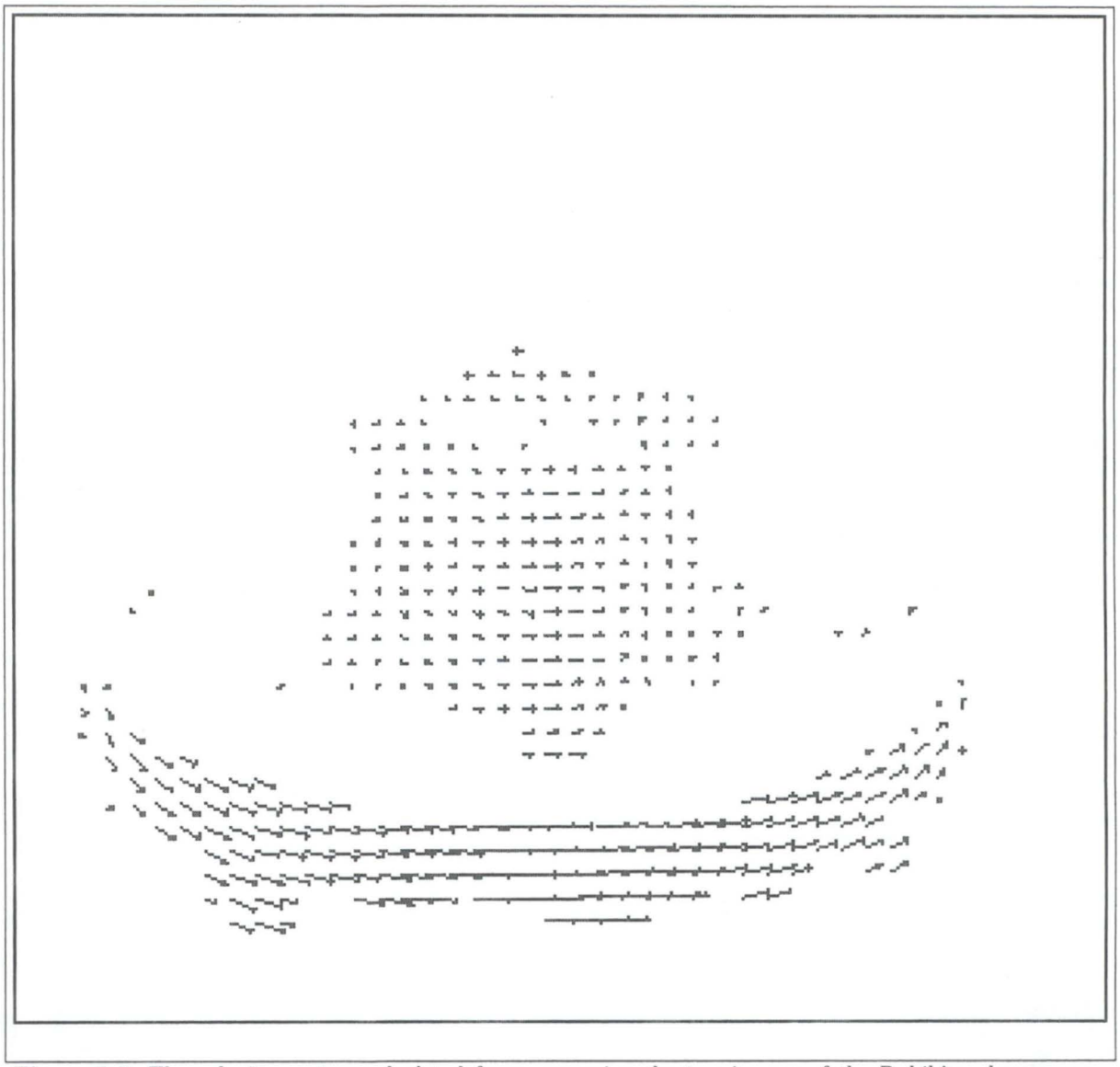


Figure 3.8: The velocity vectors calculated from comparing the two images of the Rubik's cube sequence (figure 3.7). The vectors represent the counter-clockwise rotation of the turntable and the cube. The other point of the image are white because of the static background.

The gradient-based method computes a possibly different spatial transform for each pixel. The goal of motion analysis in the case of image registration is to find a global geometric transformation which can be applied over the entire image.

For example, let consider the affine transformation which relates the coordinates of corresponding pixels (x, y) and (x', y') in the two images I_1 and I_2 as follows:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \Delta_x \\ \Delta_y \end{pmatrix} + \begin{pmatrix} sc_x & sh_x \\ sh_y & sc_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.34}$$

affine flow field is expressed as

$$\begin{aligned} \vartheta_x &= x' - x = m_0 + m_1 x + m_2 y \\ \vartheta_y &= y' - y = m_3 + m_4 x + m_5 y \end{aligned} \tag{3.35}$$

which can be rewritten

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} m_0 \\ m_3 \end{pmatrix} + \begin{pmatrix} 1+m_1 & m_2 \\ m_4 & 1+m_5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.36}$$

Thus, the parameters of a general 2D affine transformation can be completely determined by the affine 6-dimensional flow vector $\vec{m} = (m_0, m_1, m_2, m_3, m_4, m_5)^T$. Taking the matrix form of equation (3.35), we have:

$$\vec{\vartheta} \triangleq \begin{pmatrix} \vartheta_x \\ \vartheta_y \end{pmatrix} = \mathcal{X}\vec{m} \tag{3.37}$$

where

$$\mathcal{X} = \begin{pmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{pmatrix} \tag{3.38}$$

Then, to recover the parameters m_i of the vector $\vec{m}$, we rewrite the gradient constraint (3.32) as follows

$$\Big(I_x(x,y,t), I_y(x,y,t)\Big) \begin{pmatrix} \vartheta_x \\ \vartheta_y \end{pmatrix} + I_t(x,y,t) = 0 \tag{3.39}$$

Finally combining equations (3.37) and (3.39) results in

$$\Big(I_x(x,y,t), I_y(x,y,t)\Big)\mathcal{X}\vec{m} + I_t(x,y,t) = 0 \tag{3.40}$$

or more precisely

$$\begin{pmatrix} I_x(x,y,t), & xI_x(x,y,t), & yI_x(x,y,t), & I_y(x,y,t), & xI_y(x,y,t), & yI_y(x,y,t) \end{pmatrix} \vec{m} + I_t(x,y,t) = 0 \tag{3.41}$$

To obtain a global affine transformation which will be approximately valid for the whole image, we want to estimate a single set of affine parameters m_i instead of a velocity vector for each pixel. So we take into consideration all pixels inside the overlapping area[10]. The equation (3.41) is extended into a linear matrix equation as follows:

$$\mathcal{A}\vec{m} + \vec{b} = 0 \tag{3.42}$$

where each row in the matrix $\mathcal{A}$ consists of the vector in brackets in equation (3.41) for a certain pixel (x, y) and each component of the vector $\vec{b}$ consists of the right part of equation (3.41).

In practice, the intensity conservation assumption is only approximately true because of possible changes in intensity. Therefore, the problem is to minimise the Sum of Squared Differences[11]:

$$SSD = \sum_{x,y} \left(I(x + \vartheta_x dt, y + \vartheta_y dt, t + dt) - I(x,y,t)\right)^2 \tag{3.43}$$

[10] If the two images mainly overlap, then the overlapping area is over the whole images. If not (i.e. only a small part of the images overlaps), the global translation must be estimated in order to apply the gradient-based

which becomes the squared error summed over all the considered pixels according to equation (3.42):

$$SE = \sum_{x,y} (\mathcal{A}\vec{m} + \vec{b})^2 \tag{3.44}$$

3.2.3 Advantages and drawbacks

The gradient-based method can either compute a velocity vector for each pixel or estimate a global transformation which approximately aligns the two images.

The optical flow based method can easily be extended to use full 2D projective models and eliminates the need for a calibrated camera. All available information in the images is statistically optimally used. However, a first estimation of the transformation parameters is required in the case of large displacement between images e.g. very different viewpoints.

According to R. Szeliski and J. Coughlan, the main drawback of the minimisation scheme presented above is that it will typically have many locally optimal solutions. Indeed, on the one hand we have a velocity vector for each point and on the other hand we have a global estimation of the transformation if we compute a single set of parameters. In this case, it will have many local misregistrations. So they proposed a spline based image registration [SC94] which removes the need for overlapping correlation windows and produces an explicit measure of the correlation between adjacent motion estimates.

3.3 Intensity based registration

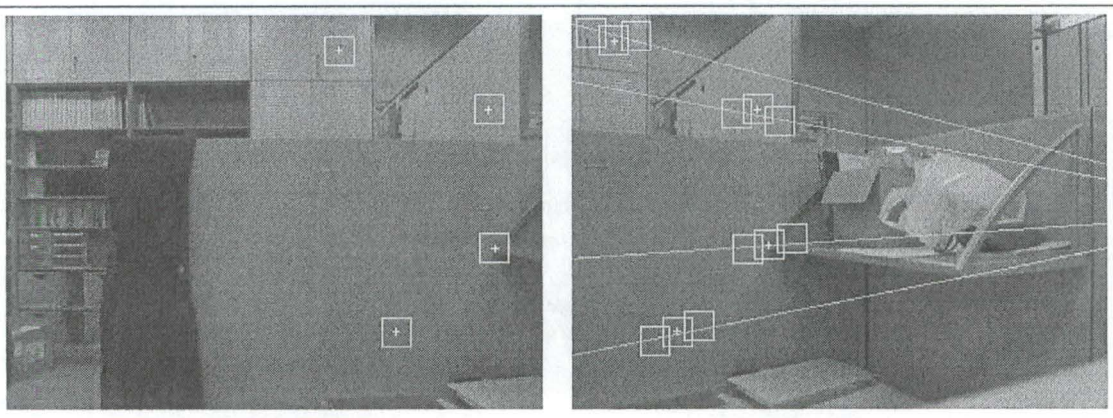
There are a lot of registration methods based on intensity i.e. the value of each pixel in the image. In fact, the optical flow based method seen before is also based on intensity. Here, we will present two well known algorithms:

- the correlation-based method found in [TV98]. This method uses a correlation window in the second image to find the correspondence with a template from the first image;
- the method based on the *Levenberg-Marquardt* iterative nonlinear minimisation algorithm. It is detailed in [Sze96].

These approaches are similar to the optical flow based method if it is only that the projected points (i.e. pixel coordinates) are directly used instead of the motion vectors.

3.3.1 The correlation-based method

For a pixel $P_1(x, y)$ in the first image, this method uses a correlation window centred on $P_1(x, y)$ (called the template) and looks for the window centred on the corresponding pixel $P_2(x + \Delta_x, y + \Delta_y)$ in the second image i.e. the window producing the highest correlation. Δ_x and Δ_y refer to the disparity search range $\mathbb{T}$ in the image I_2 for $P_1(x, y)$. This search space $\mathbb{T}$ corresponds to a set of possible translations $\mathcal{T}(\Delta_x, \Delta_y)$ along the x-axis and y-axis. $\mathbb{T}$ should be as limited as possible. For example, knowing the *Fundamental matrix* F between the cameras, the search of the corresponding pixels can be reduced to a 1D space: the epipolar line[12] of $P_1(x, y)$ (see figure 3.9).



(a) The left image I_1 (b) The right image I_2

Figure 3.9: Assuming the left picture 3.9(a) to be the reference image and the other picture 3.9(b) to be on the right, the algorithm looks for the point in the adjacent image corresponding to the central pixel of the window in the reference image. This window is correlated to several windows of the same width $2W + 1$ in the other image (only a few are drawn here). In the adjacent image, the centre of the correlation window producing the highest correlation is the corresponding point we are looking for. In this example, the search space is restricted to the pixels in the vicinity of epipolar lines of each searched pixels $P_1(x, y)$. The corresponding pixels obtained with this method are marked by a cross.

As it is illustrated on figure 3.9, the algorithm consists of:

1. For each translation $\mathcal{T}(\Delta_x, \Delta_y) \in \mathbb{T}$, compute the Normalised Cross-Correlation[13]:

$$NCC = \frac{\sum_{i=-W}^{W}\sum_{j=-W}^{W}\Big(I_1(x+i,y+j)-\overline{I_1}\Big)\Big(I_2(x'+i,y'+j)-\overline{I_2}\Big)}{\sqrt{\sum_{i=-W}^{W}\sum_{j=-W}^{W}\Big(I_1(x+i,y+j)-\overline{I_1}\Big)^2\sum_{i=-W}^{W}\sum_{j=-W}^{W}\Big(I_2(x'+i,y'+j)-\overline{I_2}\Big)^2}} \tag{3.45}$$

   where $x' = x + \Delta_x$ and $y' = y + \Delta_y$, $\overline{I_1}$ and $\overline{I_2}$ respectively denote the average of the value of pixels in the template and in the window of the adjacent image;

2. The translation $\overline{\mathcal{T}}(\Delta_x, \Delta_y)$ is the one that maximises NCC over $\mathbb{T}$. This is also called the disparity of the pixel $P_1(x, y)$.

The size $2W + 1$ of the correlation window is a determining parameter because

- *too small* a window may not capture enough image structure and may be too noise sensitive which would produce many false matches;
- *too large* a window makes the matching less sensitive to noise but also to actual variations of image intensity which is not desired.

The success of the matching using correlation windows also depends on the presence of distinctive structure in the window that occurs infrequently in the search region of the other image.

The correlation-based method gives a pixel-by-pixel correspondence. In order to estimate a global geometric transformation $\mathcal{M}$ that relates the two images, one can simply resolve the following equations with at least four pairs of corresponding points widespread in the images:

$$\begin{aligned} x' &= \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1} \\ y' &= \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1} \end{aligned} \tag{3.46}$$

where (x, y) and (x', y') are pairs of corresponding pixels and $(m_0, \ldots, m_7)$ are the unknown factors to be estimated. This can be written as:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -x_1y_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1x_1' & -y_1y_1' \\ & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx_n' & -x_ny_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_nx_n' & -y_ny_n' \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ \vdots \\ m_6 \\ m_7 \end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \\ \vdots \\ x_n' \\ y_n' \end{pmatrix} \tag{3.47}$$

in the general case of a given set of n corresponding points coordinates.

3.3.2 Using the *Levenberg-Marquardt* algorithm

Working with intensity generally means maximising a similarity measure or minimising an error function[14]. In [Sze96], the goal is to iteratively estimate a transformation, say a projective one, by minimising the discrepancy in intensities.

Let (x_i, y_i) denote a pixel in the reference image I_1. Given an estimated[15] perspective transformation matrix $\mathcal{M}$, we can obtain the corresponding pixel (x'_i, y'_i) in the image I_2 from the pixel (x_i, y_i) in the reference image I_1 by rewriting the projective transformation as:

$$\begin{aligned} x'_i &= \frac{m_0 x_i + m_1 y_i + m_2}{m_6 x_i + m_7 y_i + 1} \\ y'_i &= \frac{m_3 x_i + m_4 y_i + m_5}{m_6 x_i + m_7 y_i + 1} \end{aligned} \tag{3.48}$$

The Szeliski's technique simply minimises the **S**um of the **S**quared intensity **E**rrors overall these corresponding pairs of pixels i (pixels that are mapped outside image boundaries do not contribute):

$$SSE = \sum_{i \in I_1 \cap I_2} \left(I_2(x'_i, y'_i) - I_1(x_i, y_i) \right)^2 = \sum_{i \in I_1 \cap I_2} e_i^2 \tag{3.49}$$

where $i \in I_1 \cap I_2$ expresses that we only consider the pixels with coordinates (x_i, y_i) which have their correspondent in the boundaries of the image I_2 (i.e. the coordinates (x'_i, y'_i) obtained with the equations 3.48 lie in I_2). However, the computed coordinates (x'_i, y'_i) do not fall on integer pixel coordinates so Szeliski suggests to use bilinear interpolation[16] to get the intensity value of $I_2(x'_i, y'_i)$.

The minimisation of the error function SSE is performed by using the *Levenberg-Marquardt* iterative nonlinear minimisation algorithm. By considering the transformation matrix $\mathcal{M}$ as a vector $\vec{m} = (m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7)^T$, this algorithm computes an approximate Hessian matrix $\mathcal{A}$ and a weighted gradient vector $\vec{b}$ from the partial derivatives of e_i. The weight of the gradient vector is simply the error term e_i. The algorithm consists of four main steps:

Step 1: For each pixel i at location (x_i, y_i)

1. Compute the corresponding coordinates (x'_i, y'_i) according to the current transformation matrix $\mathcal{M}$ i.e. vector $\vec{m}^{(t)}$ (see equations 3.48);
2. Compute the difference in intensity $e_i = I_2(x'_i, y'_i) - I_1(x_i, y_i)$;
3. Compute the partial derivatives of e_i with respect to the m_k terms using:

$$\frac{\partial e_i}{\partial m_k} = \frac{\partial I_2}{\partial x'_i} \frac{\partial x'_i}{\partial m_k} + \frac{\partial I_2}{\partial y'_i} \frac{\partial y'_i}{\partial m_k} \tag{3.50}$$

4. Add the pixel contribution to $\mathcal{A}$ and $\vec{b}$ as follows:

[14] See section 3.5 "The similarity measure".

[15] The estimated perspective transformation can be the result of the previous iteration or can come from a

$\mathcal{A}$ is a 8×8 matrix and its components are:

$$\mathcal{A}_{kl} = \sum_{i \in I_1 \cap I_2} \frac{\partial e_i}{\partial m_k} \frac{\partial e_i}{\partial m_l} \tag{3.51}$$

Whereas gradient vector $\vec{b}$ has 8 elements:

$$b_k = - \sum_{i \in I_1 \cap I_2} e_i \frac{\partial e_i}{\partial m_k} \tag{3.52}$$

Note that the partial derivatives of e_i (3.50) are straightforward to compute:

$$\frac{\partial e_i}{\partial m_0} = \frac{x_i}{D_i} \frac{\partial I_2}{\partial x'_i} \qquad \frac{\partial e_i}{\partial m_1} = \frac{y_i}{D_i} \frac{\partial I_2}{\partial x'_i}$$

$$\frac{\partial e_i}{\partial m_2} = \frac{1}{D_i} \frac{\partial I_2}{\partial x'_i} \qquad \frac{\partial e_i}{\partial m_3} = \frac{x_i}{D_i} \frac{\partial I_2}{\partial y'_i}$$

$$\frac{\partial e_i}{\partial m_4} = \frac{y_i}{D_i} \frac{\partial I_2}{\partial y'_i} \qquad \frac{\partial e_i}{\partial m_5} = \frac{1}{D_i} \frac{\partial I_2}{\partial y'_i}$$

$$\frac{\partial e_i}{\partial m_6} = -\frac{x_i}{D_i}\left(x'_i \frac{\partial I_2}{\partial x'_i} + y'_i \frac{\partial I_2}{\partial y'_i}\right) \qquad \frac{\partial e_i}{\partial m_7} = -\frac{y_i}{D_i}\left(x'_i \frac{\partial I_2}{\partial x'_i} + y'_i \frac{\partial I_2}{\partial y'_i}\right)$$

where $D_i = m_6 x_i + m_7 y_i + 1$ is the denominator in (3.48) and $\left(\frac{\partial I_2}{\partial x'_i}, \frac{\partial I_2}{\partial y'_i}\right)$ is the image intensity gradient of $I_2(x'_i, y'_i)$:

$$\nabla I_2(x'_i, y'_i) = \left(\frac{\partial I_2(x'_i, y'_i)}{\partial x'_i} \frac{\partial I_2(x'_i, y'_i)}{\partial y'_i}\right) \tag{3.53}$$

A derivative filter is used to approximate the image intensity gradient $\nabla I_2(x'_i, y'_i)$ as explained in chapter 5. For example, we can use the *Sobel* filter[17] which gives:

$$\begin{aligned} \frac{\partial I_2(x'_i, y'_i)}{\partial x'_i} &= I_2(x'_i + 1, y'_i - 1) + 2I_2(x'_i + 1, y'_i) + I_2(x'_i + 1, y'_i + 1) \\ &\quad - I_2(x'_i - 1, y'_i - 1) - 2I_2(x'_i - 1, y'_i) - I_2(x'_i - 1 y'_i + 1) \end{aligned} \tag{3.54}$$

$$\begin{aligned} \frac{\partial I_2(x'_i, y'_i)}{\partial y'_i} &= I_2(x'_i - 1, y'_i + 1) + 2I_2(x'_i, y'_i + 1) + I_2(x'_i + 1, y'_i + 1) \\ &\quad - I_2(x'_i - 1, y'_i - 1) - 2I_2(x'_i, y'_i - 1) - I_2(x'_i + 1, y'_i - 1) \end{aligned} \tag{3.55}$$

Then, the next steps of the *Levenberg-Marquardt* algorithm are:

Step 2: Solve the system of equations $(\mathcal{A} + \lambda \mathcal{I}) \Delta \vec{m} = \vec{b}$ where $\mathcal{I}$ is the identity matrix and λ is a time-varying stabilisation parameter which is initialised with a modest value[18], say 0.001. If we select $\lambda = 0$, the algorithm performs a *Newton-Gauss* minimisation. With this parameter λ, the *Levenberg-Marquardt* algorithm smoothly varies between the *Newton-Gauss* and the steepest descent method;

Step 3: Compute SSE again with $\vec{m}^{(t)} + \Delta \vec{m}$ and check if it has decreased:

- if it is so, decrease[19] λ by a factor of 10 and update the parameter vector $\vec{m}^{(t+1)} = \vec{m}^{(t)} + \Delta\vec{m}$. Note that the image I_2 must be temporarily transformed with the current parameters in order to compute the right partial derivatives during the next iteration;
- if not, the updated failed to reduce the residual, so increase[20] λ by a factor of 10 and set $\vec{m}^{(t+1)} = \vec{m}^{(t)}$.

Step 4: Continue iterating until the error is below a fixed threshold or a fixed number of iterations has been completed.

The output vector $\vec{m}$ contains the eight estimated parameters of the perspective transformation $\mathcal{M}$.

3.3.3 Advantages and drawbacks

The method using correlation windows can be used to obtain a first estimation of the geometric transformation. However, the pairs of corresponding points must be quite selected to be representative of the distortion between the images (e.g. the four selected points in the first image are the four corners of a rectangle which covers the overlapping area). In addition, the correlation of the template and windows is not accurate when the viewpoints of the cameras are very different i.e. when the geometric distortion is important between the images.

The *Levenberg-Marquardt* approach uses image pixel values directly, and minimises the discrepancy of intensities between a warped image and the reference image by statistically using all the information. According to Szeliski, the advantage of using *Levenberg-Marquardt* over straightforward gradient descent is that it converges in fewer iterations. The major drawback is that it is very sensitive to geometric distortion and it requires a good initial approximation of the geometric transformation to register images with large distortion.

In addition to the computational cost of temporary transformations and interpolation, this method is sensitive to intensity variations due to video camera gain control and illumination scene changes. In [JC01], they use multiresolution by first estimating the translation at the coarsest level, then refining an affine transformation and at the finest level, they estimate the projective transformation initialised with the affine parameters. Beyond the projective model, they also introduce a polynomial illumination change model in the algorithm above to handle the problem of illumination change.

[19] The parameter λ should approach 0 as the minimum is achieved so that the influence of the diagonal

3.4 Feature based registration

Feature based methods rely on three main steps:

1. Locate and extract a number of features in both images;
2. Match the common main features from the images;
3. Compute the parameters of the transformation from the features location.

Several types of features have been used to accomplish the registration, frequently used ones include points, lines and polygons. According to the chosen features, the methods will differ in the two first steps, the third one depending on the class of transformations which is considered.

In her survey [Bro92], L. G. Brown calls this method *Point Mapping* and distinguishes the intrinsic and the extrinsic RCPs[21]. The intrinsic RCPs are some kind of markers placed specifically for the registration task and are not part of the image data itself. Although these points can be easily identified and make the registration easier, it is not always possible to resort to these artefacts. The extrinsic RCPs belong to the image data itself and can be manually selected or automatically found.

Corners, line intersections as well as points of locally maximum curvature on contour lines are commonly used as relevant control points, the principle being to select points which are likely to be uniquely found in both images and more tolerant of local distortions. On the one hand, the number of control points in both images should be large enough to ensure a sufficient number of pairs of corresponding points which will be used in the computing of the transformation. On the other hand, the higher the number of control points, the more difficult the matching step.

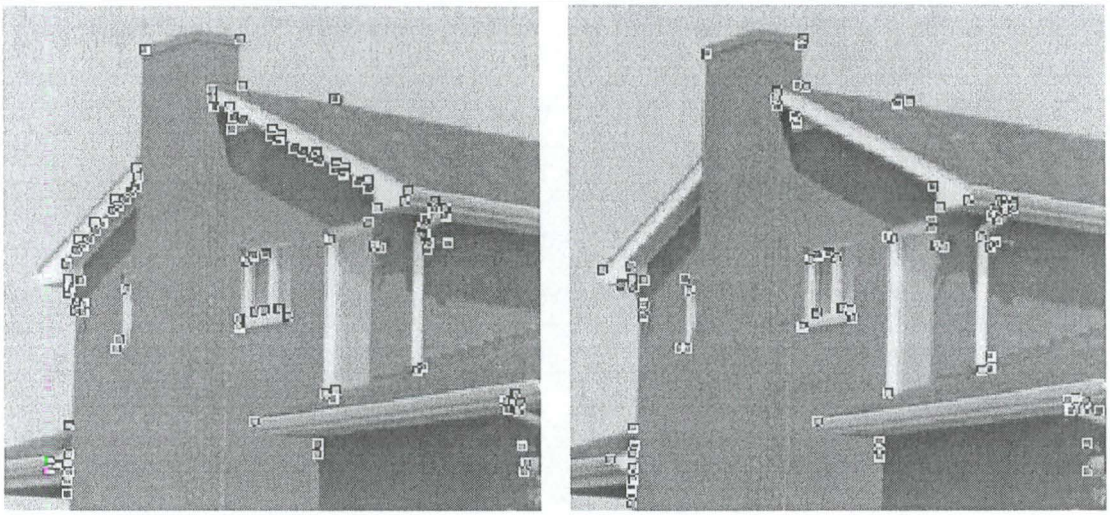
In what follows, we only present image registration using corners which is commonly used. So we first explain how to extract corners in an image. Then the matching of corners and the estimation of the transformation is presented in the case of two overlapping images.

3.4.1 The extraction of corners

The most intuitive type of feature point is the corner which is commonly defined in the image processing field as [TH98]: «*Corners are image points that show a strong two dimensional intensity change, and are therefore well distinguished from neighbouring points*». M. Trajković and M. Hedley in [TH98] classify the corner detectors in two main classes:

- curvature based;
- "interest operators" or feature point detectors.

In what follows, only one method of each class is presented, namely the Kitchen & Rosenfeld corner detector and the Harris corner detector. On figure 3.10, we can see a comparison of the corners extracted with each one of them.



(a) The Harris corner detector (b) The Kitchen & Rosenfeld corner detector

Figure 3.10: While the Harris corner detector (on figure 3.10(a)) extracts a lot of corners, the corners detected by the Kitchen & Rosenfeld approach are mainly situated at the end of edges as illustrated in figure 3.10(b).

The first class of methods identifies corners as points on the boundary of two regions of the image where the boundary curvature is sufficiently high. For example, the approach described by Kitchen and Rosenfeld defines "*cornerness*" as the product of the local gradient[22] magnitude and the rate of change of the gradient direction along an edge contour. In other words, it locates corners of edges as the local extrema of the following operator K:

$$K = \frac{\frac{\partial^2 I}{\partial x \partial x}\left(\frac{\partial I}{\partial y}\right)^2 + \frac{\partial^2 I}{\partial y \partial y}\left(\frac{\partial I}{\partial x}\right)^2 - 2\frac{\partial^2 I}{\partial x \partial y}\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}}{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{3.56}$$

where $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, $\frac{\partial^2 I}{\partial x \partial x}$, $\frac{\partial^2 I}{\partial y \partial y}$ and $\frac{\partial^2 I}{\partial x \partial y}$ denote the first and second order derivatives of image intensity. Note that K explicitly represents the second directional derivatives in the direction orthogonal to the gradient. Indeed, the partial derivatives of the gradient direction θ are:

$$\theta_x = \frac{\frac{\partial^2 I}{\partial x \partial y}\frac{\partial I}{\partial x} - \frac{\partial^2 I}{\partial x \partial x}\frac{\partial I}{\partial y}}{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \qquad \theta_y = \frac{\frac{\partial^2 I}{\partial y \partial y}\frac{\partial I}{\partial x} - \frac{\partial^2 I}{\partial x \partial y}\frac{\partial I}{\partial y}}{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{3.57}$$

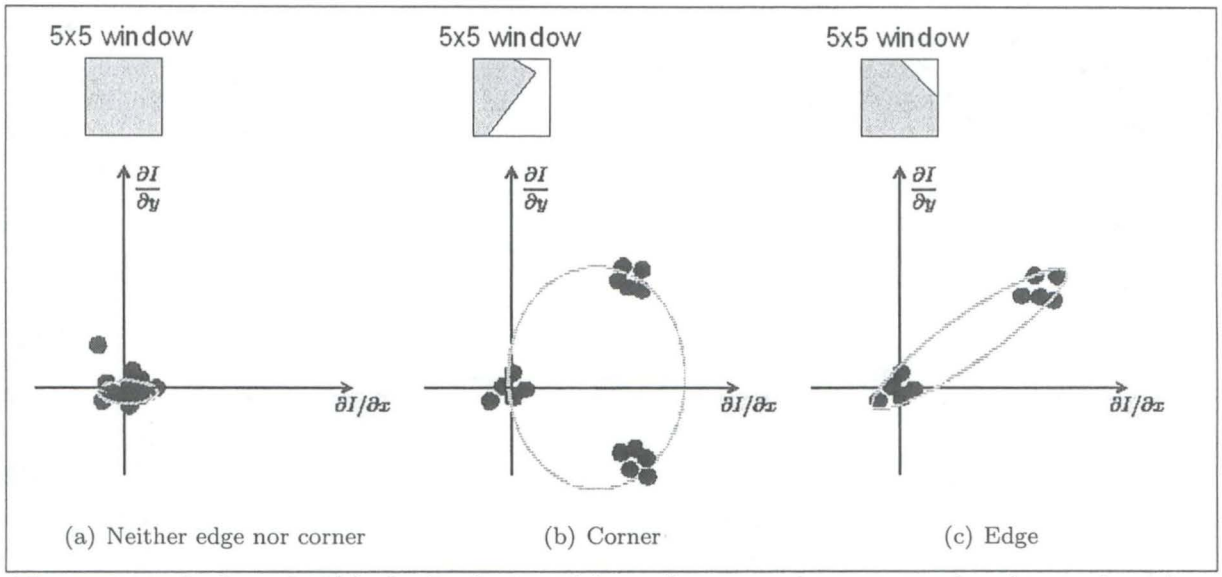
In addition, as the gradient is directed across the edge, the vector $(-\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x})^T$ is directed along the edge. Thus the measure of the cornerness K is the projection of the rate of change of the gradient direction along the edge multiplied by the gradient magnitude. An example of this corner detector is shown in figure 3.10(b).

The second class of methods redefines corners as points that are sufficiently different from their neighbours. The most widely used one, the *Plessey* feature point detector (also simply

called the *Harris* corner detector), belongs to this class and was proposed by C. Harris and M. Stephens. Consider the following matrix:

$$\mathcal{M}_P = \begin{pmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{pmatrix} \tag{3.58}$$

For a pixel P, $\mathcal{M}_P$ is formed with its neighbourhood (e.g. a 5×5 window centred on P) and is the covariance matrix of all the gradient vectors $\nabla I(x, y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)^T$ in the neighbourhood. A corner is detected if the two eigenvalues of the matrix $\mathcal{M}_P$ are large, which corresponds to



(a) Neither edge nor corner (b) Corner (c) Edge

Figure 3.11: On figure 3.11(a), the distribution of the gradient vector is compact and so the two eigenvalues are small. Consequently, it corresponds to a part of the image where the intensity is uniform. On the contrary, when the distribution is widespread as on figure 3.11(b) and results in two large eigenvalues, we can deduce that it is a corner. On the last figure (3.11(c)), if only one of the eigenvalues is large due to a distribution in one direction, and the other eigenvalue is above a certain threshold, the pixel is supposed to belong to an edge.

an important change of grey level for a small motion in any direction. Indeed, the eigenvalues represent the major and minor axis of the elliptical approximation of the gradient vector distribution (see figure 3.11). As it is underlined in [Tao02], we can also identify edges. If the smaller eigenvalue of the matrix $\mathcal{M}_P$ is larger than a certain threshold, the pixel P belongs to an edge (see figure 3.11(c)).

To avoid the eigenvalue decomposition of the matrix $\mathcal{M}_P$, corners can be defined as local maxima of the *cornerness* function R:

$$R = \det \mathcal{M}_P - k(\text{trace } \mathcal{M}_P)^2 \tag{3.59}$$

where k is a parameter set to 0.04 according to C. Harris. On figure 3.12, an example illustrates the Harris corner detector.

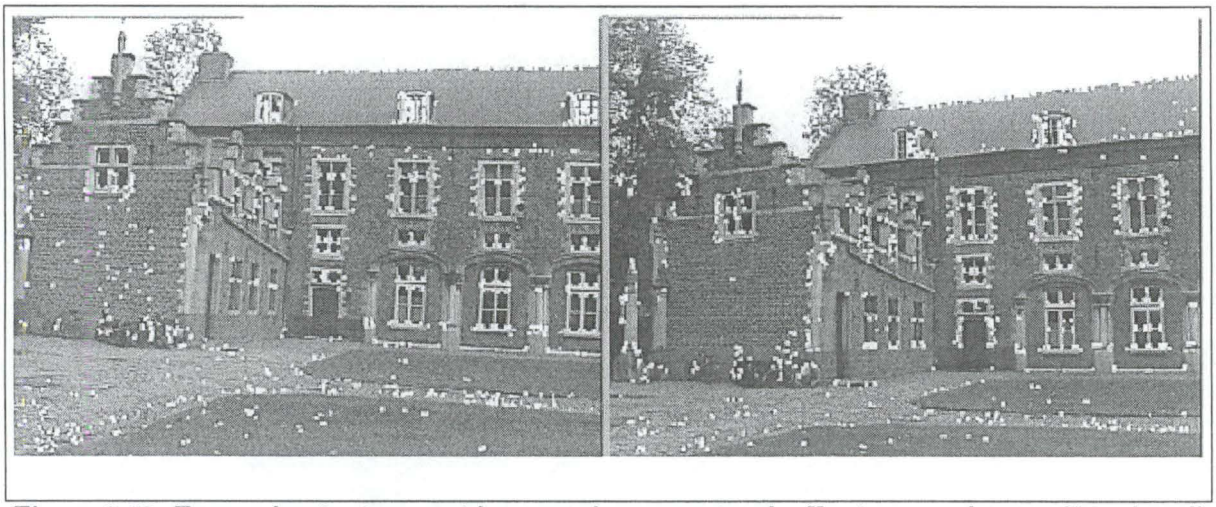


Figure 3.12: Two overlapping images with extracted corners using the *Harris corner detector*. Note that all the corners detected in an image do not necessarily have their corresponding point in the other image.

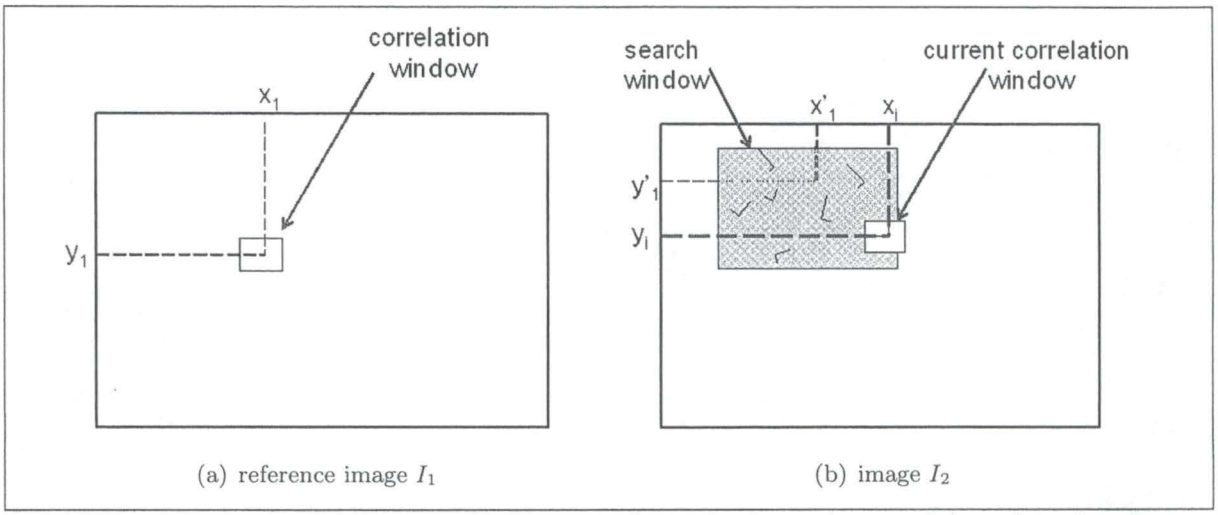
derivatives to smooth the images. Another interesting remark is that there are often too much extracted corners in practice. One possibility is to set a threshold which determines the minimum requested variation of brightness around the pixel and select only the corners with a value R above this threshold.

There also exists some others well known corner detectors such as the straightforward method SUSAN introduced by Smith and Brady, the method developed by Wang and Brady, the simple and fast detector of Trajković and Hedley,... They also propose to consider the pixel with high gradient, only to reduce the computational cost of the *Harris* corner detector. See [TH98] for explanations and comparisons of these methods.

3.4.2 The matching of corners

After the set of corners has been determined in the two images, the features in each picture must be matched. The easiest and most intuitive approach is to take a small region of pixels (referred to as a correlation window[23]) from around the detected corner with coordinates (x_1, y_1) in the reference image I_1 and compare this with a similar region from around each of the candidate corners (x_i, y_i) in the other image I_2. The search space can be restricted by defining a search window in I_2 with a radius larger than the expected displacement of the feature between the two images. Thus, all corners lying in the search window are candidates for the match (see figure 3.13).

To determine if two corners are matching, a score called the similarity measure[24] is associated to each comparison. The Normalised Cross-Correlation (3.60) is often used. The objective is to find the pair of correlation windows that maximises the NCC measure in order



(a) reference image I_1 (b) image I_2

Figure 3.13: The matching step through correlation. Given the detected corner $I_1(x_1, y_1)$ in the reference image 3.13(a), we want to find the corresponding corner $I_2(x'_1, y'_1)$ in the other image 3.13(b). A correlation window is used and the search is restricted to the search window in I_2. All the detected corners $I_2(x_i, y_i)$ inside the search window are candidates for the match.

to locate the two corresponding corners.

$$NCC = \frac{\sum_{k=-W}^{W}\sum_{l=-W}^{W}\Big(I_1(x_1+k, y_1+l) - \overline{I_1}\Big)\Big(I_2(x_i+k, y_i+l) - \overline{I_2}\Big)}{\sqrt{\sum_{k=-W}^{W}\sum_{l=-W}^{W}\Big(I_1(x_1+k, y_1+l) - \overline{I_1}\Big)^2 \sum_{k=-W}^{W}\sum_{l=-W}^{W}\Big(I_2(x_i+k, y_i+l) - \overline{I_2}\Big)^2}} \tag{3.60}$$

where

- (x_1, y_1) and (x_i, y_i) are the coordinates of the corners;
- $\overline{I_1}$ and $\overline{I_2}$ respectively denote the average of the value of pixels lying in the correlation windows of the images I_1 and I_2;
- W is related to the size of the $(2W+1) \times (2W+1)$ correlation windows.

According to [Bro92], some techniques map a set of points in one image onto the corresponding set in the second image instead of working corner by corner. Moreover, other ones determine the spatial transformation between the images while performing the matching task. This is the case of the clustering technique which tackle the problem of rotation, scaling and translation but it can be extended to other transformations. For each possible pair of matching corners, the transformation is determined which represents a point in the cluster space. By finding the best cluster of these points, the transformation which most closely matches the largest number of points is found.

3.4.3 The estimation of the transformation

of the transformation can be estimated by solving a system of equations. If we are estimating a projective transformation with 8 parameters $m_0, \ldots, m_7$, the system is rewritten in matrix notation as follows:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -x_1y'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1x'_1 & -y_1y'_1 \\ & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx'_n & -x_ny'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_nx'_n & -y_ny'_n \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ \vdots \\ m_6 \\ m_7 \end{pmatrix} = \begin{pmatrix} x'_1 \\ y'_1 \\ \vdots \\ x'_n \\ y'_n \end{pmatrix} \tag{3.61}$$

3.4.4 About other types of features

Concerning edge based registration, there are less frameworks in this field. As regards the detection of the edges, you can refer to chapter 5 "Edge detection". A complete and interesting edge based method is presented in [HLF+97] which uses a multiresolution wavelet transform to extract the features. The wavelets are the partial derivatives of a smoothing function and are decomposed into two independent components that are equivalent to the gradient of the smoothed image. They also propose a new method to eliminate the incorrectly matched pairs based on the idea that the distance between two points in the same image is preserved when it undergoes a rigid transform. This method is summarised in [HB00].

In [CL99], they present a strategy to initialise the registration using polygon features matching. They combine three descriptors for polygons, namely, *Shape-Matrix*, *Fourier Descriptors* and *Invariant Moments* in the matching scheme.

3.4.5 Advantages and drawbacks

The feature based methods select a few corresponding RCPs on the two images and estimate the parameters of the transformation using these reliable points only, which minimises the search space. As stated in [Bro92], these methods are less sensitive to local distortions because they use RCPs, area based similarity measure and the information from spatial relationships between RCPs in the images.

The critical part is the selection of features and their matching. On the one hand, this method becomes unstable and inaccurate for a large number of control points and it also requires a good correspondence between corners. On the other hand, the corresponding corners used in the estimation of the transformation parameters must not lie in the same area of the images but have to be quite widespread in the images. In addition, the extraction of characteristic features may be computationally expensive. These problems can be overcome by using more efficient and robust algorithms as suggested in [Bro92] or by using a *branch-and-bound* approach combined with computing point alignments to accelerate the search [MNLM99].

In order to handle more geometric transformations than rigid-body model, more robust mechanisms are needed such as the multiresolution wavelet approach or locally warped correlation windows according to an estimation of the relationship between the two images.

3.5 The similarity measure

The objective is to compute a measure which quantifies the quality of the matching for two given functions $I_1(x,y)$ and $I_2(x',y')$. These two functions give the respective pixel intensity values at each location (x,y) in the first image and at each location (x',y')[25] in the second image.

Note that error metrics or similarity measures aren't suited to RGB color images but to gray-scale images. So we simply add the intensities of the three colour components for each pixel as:

$$I(x,y) = 0.212671\ I(x,y)_R + 0.715160\ I(x,y)_G + 0.071169\ I(x,y)_B \tag{3.62}$$

This weighted sum of the three colour components implies a loss of useful information but reduces the complexity and the computation time of the measure.

In the case of a template registration like in the correlation-based method presented in section 3.3.1, the $(2W+1)\times(2W+1)$ pixels inside the template and those inside the correlation windows of the adjacent image can be seen as vectors $\overrightarrow{I_1}$ and $\overrightarrow{I_2}$. Then the goal is to minimise the squared Euclidean distance (3.63) between $\overrightarrow{I_1}$ and $\overrightarrow{I_2}$ to find the corresponding windows. This distance is also known as the Sum of Squared Differences:

$$SSD = \|\overrightarrow{I_1} - \overrightarrow{I_2}\|^2 = \sum_{i=-W}^{W}\sum_{j=-W}^{W}\Big(I_1(x+i,y+j) - I_2(x'+i,y'+j)\Big)^2 \tag{3.63}$$

where the sum is over all the pixels in the region of interest i.e. the template and the window.

The term $\sum_{i=-W}^{W}\sum_{j=-W}^{W} I_1^2(x+i,y+j)$ only depends on the template and is constant in the expansion of SSD:

$$SSD = \sum_{i=-W}^{W}\sum_{j=-W}^{W}\Big(I_1^2(x+i,y+j) - 2I_1(x+i,y+j)I_2(x'+i,y'+j) + I_2^2(x'+i,y'+j)\Big) \tag{3.64}$$

If the term $\sum_{i=-W}^{W}\sum_{j=-W}^{W} I_2^2(x'+i,y'+j)$ is approximately constant then the remaining Cross-Correlation term (3.65) can be used as a similarity measure between the template and the correlation windows in the adjacent image, the maximum value of CC being the best match.

$$CC = <\overrightarrow{I_1}, \overrightarrow{I_2}> = \sum_{i=-W}^{W}\sum_{j=-W}^{W} I_1(x+i,y+j)I_2(x'+i,y'+j) \tag{3.65}$$

As it is underlined in [Lew95], CC has several weaknesses in the measure of the similarity for template matching:

- matching can fail if the image energy $\sum_{i=-W}^{W}\sum_{j=-W}^{W} I_2^2(x'+i,y'+j)$ varies with position. For instance, the correlation between the template and a bright spot may be better than the right matching region in the image;

- the range of CC is dependent on the size of the template;
- CC is not invariant to illumination changes.

Thus, CC is usually normalised to unit length by dividing it by the standard deviation of both I_1 and I_2, yielding a cosine-like correlation coefficient called the **Normalised Cross-Correlation**:

$$NCC = \frac{\sum_{i=-W}^{W}\sum_{j=-W}^{W}\left(I_1(x+i,y+j)-\overline{I_1}\right)\left(I_2(x'+i,y'+j)-\overline{I_2}\right)}{\sqrt{\sum_{i=-W}^{W}\sum_{j=-W}^{W}\left(I_1(x'+i,y'+j)-\overline{I_1}\right)^2\sum_{i=-W}^{W}\sum_{j=-W}^{W}\left(I_2(x'+i,y'+j)-\overline{I_2}\right)^2}} \quad (3.66)$$

where $\overline{I_1}$ and $\overline{I_2}$ respectively denote the average of the value of pixels in the template and in the window of the adjacent image. Like SSD, NCC measures the degree of linearity between the samples being compared, the absolute value of NCC lying between 0 and 1 (1 indicating perfect matching windows).

Although NCC is preferable since it is invariant to linear brightness and contrast variations between perfect matching windows, it relies on the same strong assumptions than SSD, namely:

- the image noise is additive and Gaussian;
- there is no rotation in the image nor in the 3D space;
- there is no scaling nor perspective distortions.

These three hypothesis restrict the use of both measures because of the following deficiencies due to different viewpoint of cameras:

- in the presence of noise which is often more complicated than a Gaussian process, the line of regression obtained by the measures can be unsatisfactory;
- parts of the scene are visible in only one of the two images (occlusion and projective distortion). In this presence of depth discontinuities, only part of the data is valid for cross correlation and the outliers should be detected and discarded;
- in the presence of specular reflection, the location of highlights varies with respect to the texture and intensities at corresponding points may not be identical or even linearly related. Thus, SSD and NCC could be poor estimator of correspondence.

In order to overcome these limitations, robust statistical methods have been developed, using weights that cause outliers to contribute less weight compared to inliers. Such a method looks like

$$E = \sum_{i=-W}^{W}\sum_{j=-W}^{W}\left(I_1(x+i,y+j)-I_2(x'+i,y'+j)\right)^2 \quad (3.67)$$

where ρ_σ is a robust estimator which assigns small weights for constraints with large residue e_{ij}. For example the *Lorentzian* estimator (3.68) has been used for motion estimation and automatic satellite image registration [JC01].

$$\rho_\sigma(e_{ij}) = \log\left(1 + \frac{1}{2}\left(\frac{e_{ij}}{\sigma}\right)^2\right) \tag{3.68}$$

where σ is a threshold. In general, this kind of estimators have a parameter expressing the point at which data must be considered outliers. The value assigned to this parameter is a critical choice:

- the measure E can behave like SSD in the case of high value for the parameter σ;
- too low a value σ may cause mismatches because the influence of valid data would be reduced correspondingly.

Further, it can vary with scene depending on the image contrast and noise level.

When all the pixels of overlapping area are taking into consideration, the similarity measures must be adapted for a variable number of pixels. An obvious technique is to average the SSD for one pixel. It is known as the **M**ean **S**quared **E**rror:

$$MSE = \frac{SSD}{N} = \frac{\sum_{x,y \in \mathcal{C}} \left(I_1(x,y) - I_2(x',y')\right)^2}{N} \tag{3.69}$$

where the sum is over all the pixels in the region of interest (i.e. the current overlapping area $\mathcal{C}$) and N is the total number of pixels in the overlapping area.

MSE works well to compare various image compression techniques by replacing I_1 and I_2 respectively with the original image and the compressed version and by cumulating over the whole image. However, it isn't reliable in the case of images with variation in illumination caused by the different viewpoints of the cameras. Indeed, the larger the overlapping area is, the higher the mean value will be because of the cumulative formulation of the squared intensity errors. To overcome this problem, some other formulas are proposed in the literature. In our implementation, we use the formula S (3.70) proposed in [ZB01] and derived from the expansion of SSD (3.64):

$$S = \frac{2 \sum_{x,y \in \mathcal{C}} I_1(x,y) I_2(x',y')}{\sum_{x,y \in \mathcal{C}} \left(I_1(x,y)^2 + I_2(x',y')^2\right)} \tag{3.70}$$

where $\mathcal{C}$ is the current overlapping area. This measure S is comprised between 0 and 1.

3.6 Conclusion

The different approaches discussed in this chapter are not exclusive. Many developed algorithms combine several techniques to overcome the problems of a single one. For example, when the displacement between two images is large, the phase correlation is often used to get an initial approximation before applying an intensity based algorithm. Another common improvement is the multiresolution matching (also called hierarchical matching). It consists of registering smaller, subsampled versions of the images and then refining the results on higher-resolution up to the original full resolution images. This approach commonly refers to a pyramid where the bottom is the original image and the top the coarsest level. As in [ZB01] or in [JC01], different methods are used according to the level of the pyramid.

Other methods use some heuristics to avoid getting stuck in various local minima[26] such as threshold, weight, limit number of iterations,... But these heuristics must be tailored to the type of source images and cannot be used in general cases.

Chapter 4

Triangulation

Following [Ioc98] the 3D position (X, Y, Z) of a point X can be reconstructed from the perspective projection of X on the image planes of two cameras, once the relative position and orientation of the two cameras are known. This chapter will explain simply the model used to achieve a scene point reconstruction which is called triangulation.

4.1 A review of the camera model

We suppose that a 3D point X is visible in an image, let X_p be the projection of X in this image. The camera is supposed to be a pinhole camera[1], thus the relation between the world coordinates of a point $X(X, Y, Z)$ and the coordinates on the image plane $X_p(x, y)$ is

$$\begin{aligned} x &= f\frac{X}{Z} \\ y &= f\frac{Y}{Z} \end{aligned} \tag{4.1}$$

where f is the focal distance of the lens and Z is the coordinate along the optical axis(see figure 4.1).

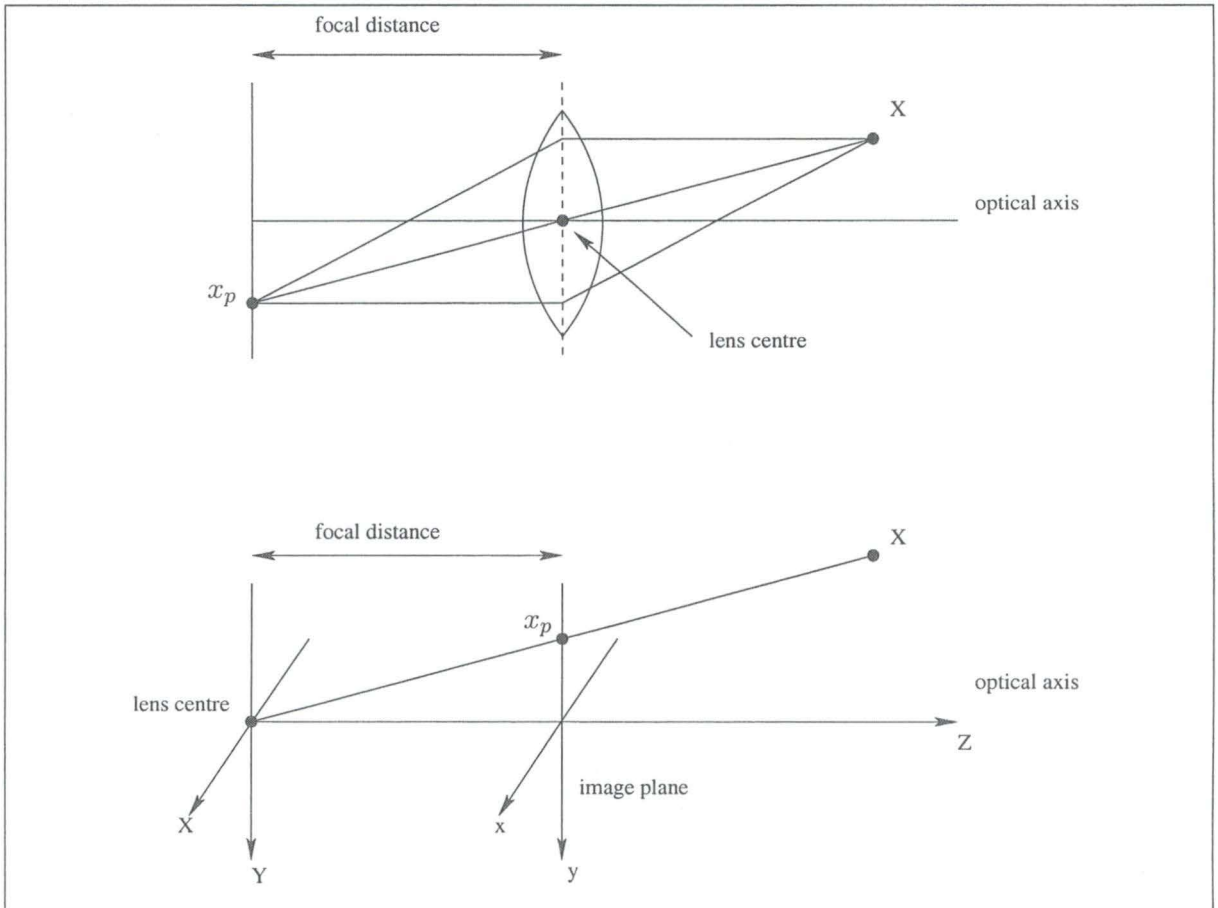


Figure 4.1: A review of the pinhole camera model.

4.2 From frame to image coordinates

A digitalised image is usually stored in a frame buffer, that can be seen as a matrix of pixels with N columns and M rows[2]. Let (i, j) be the discrete frame coordinates of the image with origin in the upper left corner, (O_x, O_y) be the principal point (i.e. the intersection between the optical axis and the image plane) in the frame coordinates, and (x, y) be the image coordinates corresponding to (i, j).

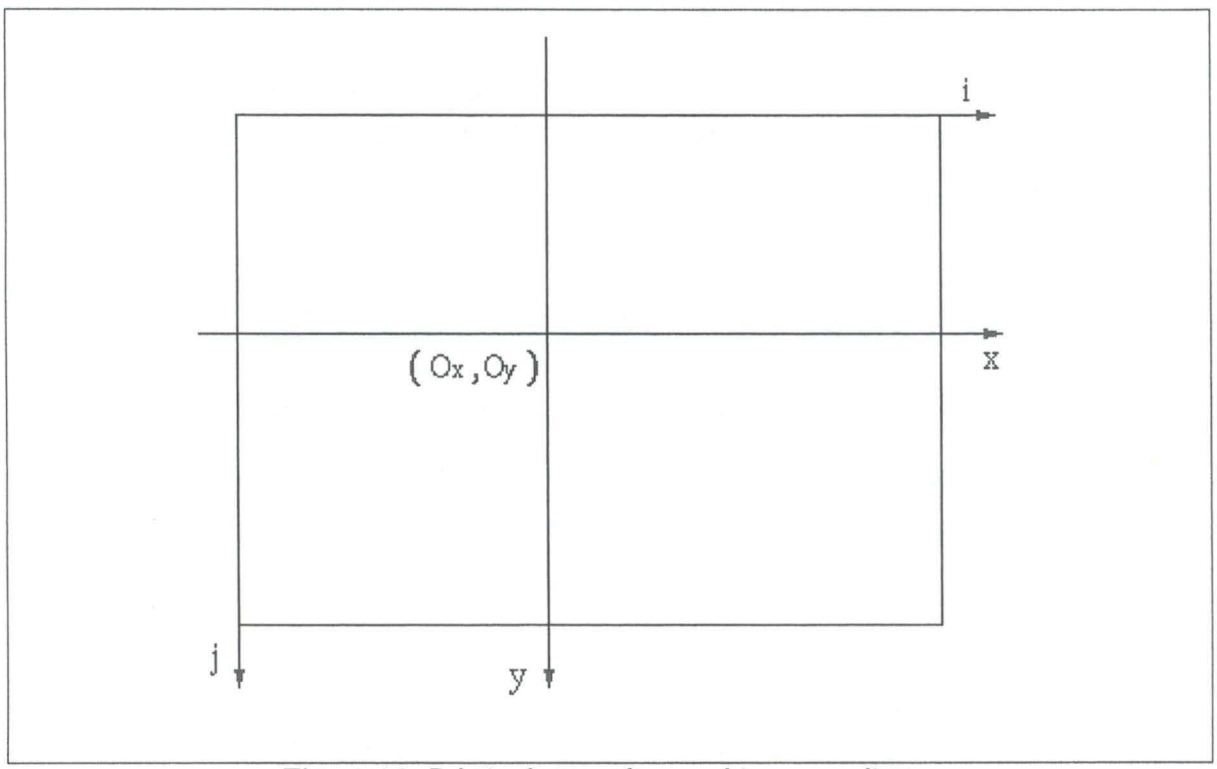


Figure 4.2: Relation between frame and image coordinates

Image coordinates relate to frame coordinates in the following way:

$$\begin{aligned} x &= (i - O_x)\, S_x \\ y &= (j - O_y)\, S_y \end{aligned}$$

where S_x and S_y are the horizontal and the vertical distances of two adjacent pixels in the frame buffer.

4.3 Triangulation

The principle is relatively simple: knowing the position of the optical centre of an image and a point from the image plane, we can "draw" a line through these two points. If we do the same for the second image, the two lines will intersect and this intersection will be the 3D point.

To make it easier, we choose the 3D world reference system to be the left camera reference system. The right camera is translated and rotated with respect from the left one, so six parameters describe this transformation.

The simplest case arises when the optical axes of the cameras are parallel, and the translation of the right camera is along the X axis.

4.3.1 General case

Given the translation vector $\mathcal{T}$ and the orthonormal rotation matrix $\mathcal{R}$ describing the transformation from left camera to right camera coordinates, the equation to solve for stereo triangulation is

$$X' = \mathcal{R}^T(X - \mathcal{T}) \tag{4.2}$$

where $X = (X, Y, Z)$ and $X' = (X', Y', Z')$ are the coordinates of the 3D point X in the left and right camera coordinates respectively, and $\mathcal{R}^T$ is the transposed matrix of $\mathcal{R}$.

4.3.2 The case of parallel cameras (perpendicular to their baseline)

Let us consider the optical setting in the figure 4.3, that is also called the *standard model.*

- Hypotheses:

  1. L and R are two pinhole cameras with parallel optical axes and the same focal length f.
  2. The baseline (i.e. the line connecting the two lenses/optical centres) is perpendicular to the optical axes. Let b be the distance between the two lens centres.

- Notations:

  1. XZ is the plane where the optical axes lie, XY plane is parallel to the image plane of both cameras, the X axis is the baseline and the origin O of (X, Y, Z) world reference system is the lens centre of the left camera (L).
  2. (x_1, y_1) is the image in the left camera of the point X and (x_2, y_2) the image in the right camera.

In this setting the equations (4.2) of stereo triangulation becomes

$$X' = \mathcal{I}(X - \mathcal{T}) \tag{4.3}$$

where $\mathcal{I}$ is the identity matrix. This system can be written in matrix form

$$\begin{pmatrix} X' \\ \end{pmatrix} = \left(\begin{pmatrix} X \\ \end{pmatrix} - \begin{pmatrix} b \\ \end{pmatrix} \right)$$

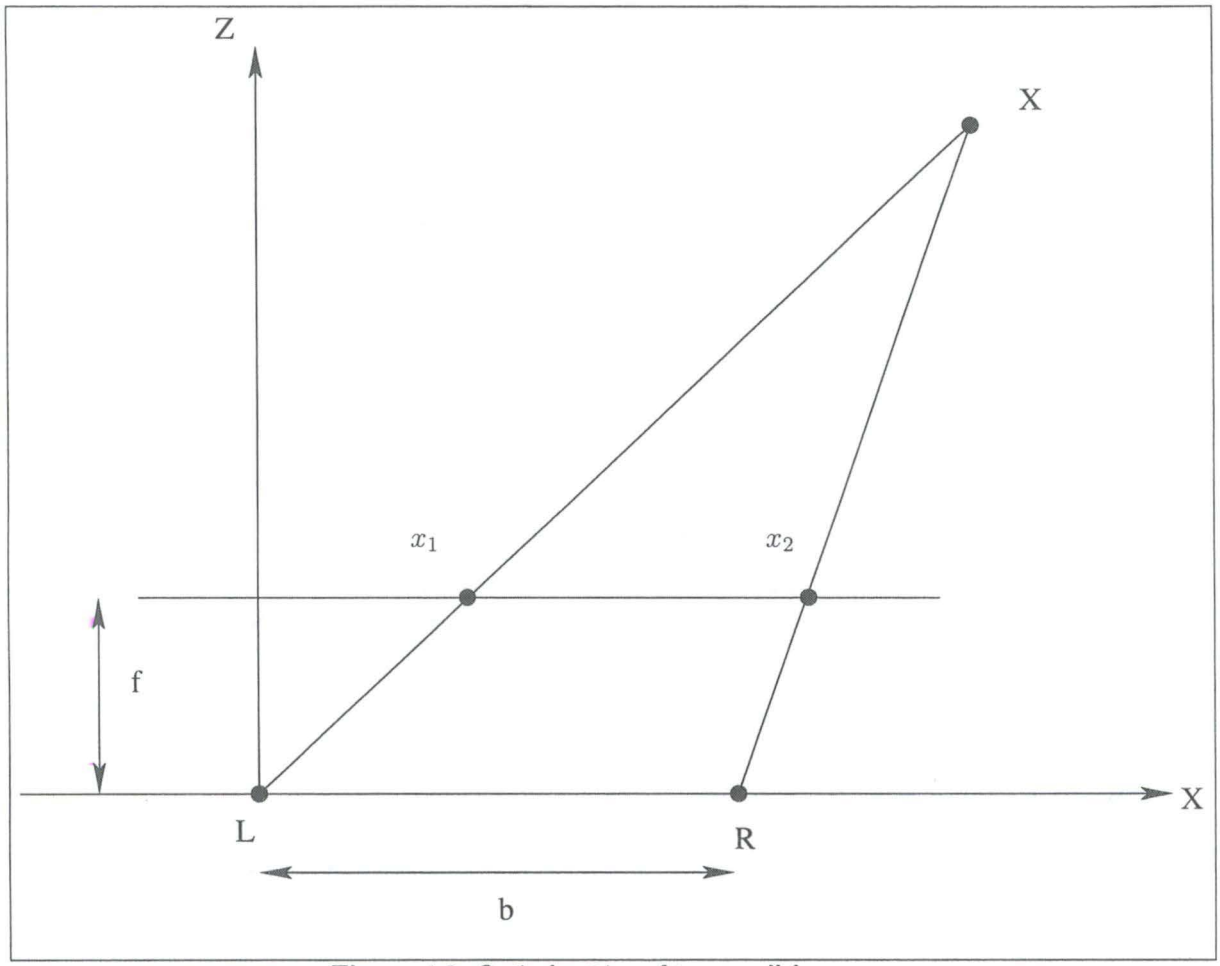


Figure 4.3: Optical setting of two parallel cameras

The setting described in figure 4.3 allows us to rewrite the translation vector $\mathcal{T}$ as $(b, 0, 0)$. Furthermore, the equations (4.1) give

$$\begin{aligned} X &= x_1 \frac{Z}{f} \\ Y &= y_1 \frac{Z}{f} \\ X' &= x_2 \frac{Z'}{f} \\ Y' &= y_2 \frac{Z'}{f} \end{aligned} \tag{4.5}$$

where

- (x_1, y_1) and (x_2, y_2) are the position of the projection of X in the left and right image coordinate respectively.

Thus, the equation (4.4) becomes

$$\begin{pmatrix} x_2\frac{Z'}{f} \\ y_2\frac{Z'}{f} \\ Z' \end{pmatrix} = \begin{pmatrix} x_1\frac{Z}{f} - b \\ y_1\frac{Z}{f} \\ Z \end{pmatrix} \tag{4.6}$$

This system can be solved easily and we obtain the solution

$$\begin{aligned} Z &= \frac{(bf)}{(x_1 - x_2)} \\ X &= x_1\frac{Z}{f} \\ Y &= y_1\frac{Z}{f} \end{aligned}$$

4.3.3 Slightly non-parallel cameras

In the general case, the right camera can be rotated with respect to the left one. This new setting will complicate the triangulation. For example we will discuss the case of a rotation around the Y axis.

Rotation around the Y axis (θ)

In this case, the equation (4.2) can be rewritten as

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \left(\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} - \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \right) \tag{4.7}$$

Again, using the equations (4.5) this system can be rewritten and we can finally use

$$\begin{aligned} (\cos\theta\frac{x'}{f} + \sin\theta)Z' - \frac{x}{f}Z + t_1 &= 0 \\ \frac{y'}{f}Z' - \frac{y}{f}Z + t2 &= 0 \\ (-\sin\theta\frac{x'}{f} + \cos\theta)Z' - Z + t_3 &= 0 \end{aligned} \tag{4.8}$$

The resolution of the system gives us the following solutions

$$\begin{aligned} Z' &= \frac{\frac{x}{f}t_3 - t_1}{\left((\cos\theta\frac{x'}{f} + \sin\theta) - \frac{x}{f}(-\sin\theta\frac{x'}{f} + \cos\theta)\right)} \\ Z &= (-\sin\theta\frac{x'}{f} + \cos\theta)Z' + t_3 \end{aligned} \tag{4.9}$$

To illustrate this case like in the parallel case, we will simplify and assume that the rotation angle are small (*small angle approximation*). Thus we can still assume that the right image plane is parallel to the left image plane and hence to XY plane, so the difference between the two images is an horizontal translation. In this case the optical axes are not parallel, but they both lie on the XZ plane, so they intersect in a point $(0, 0, Z_c)$. This point is called the

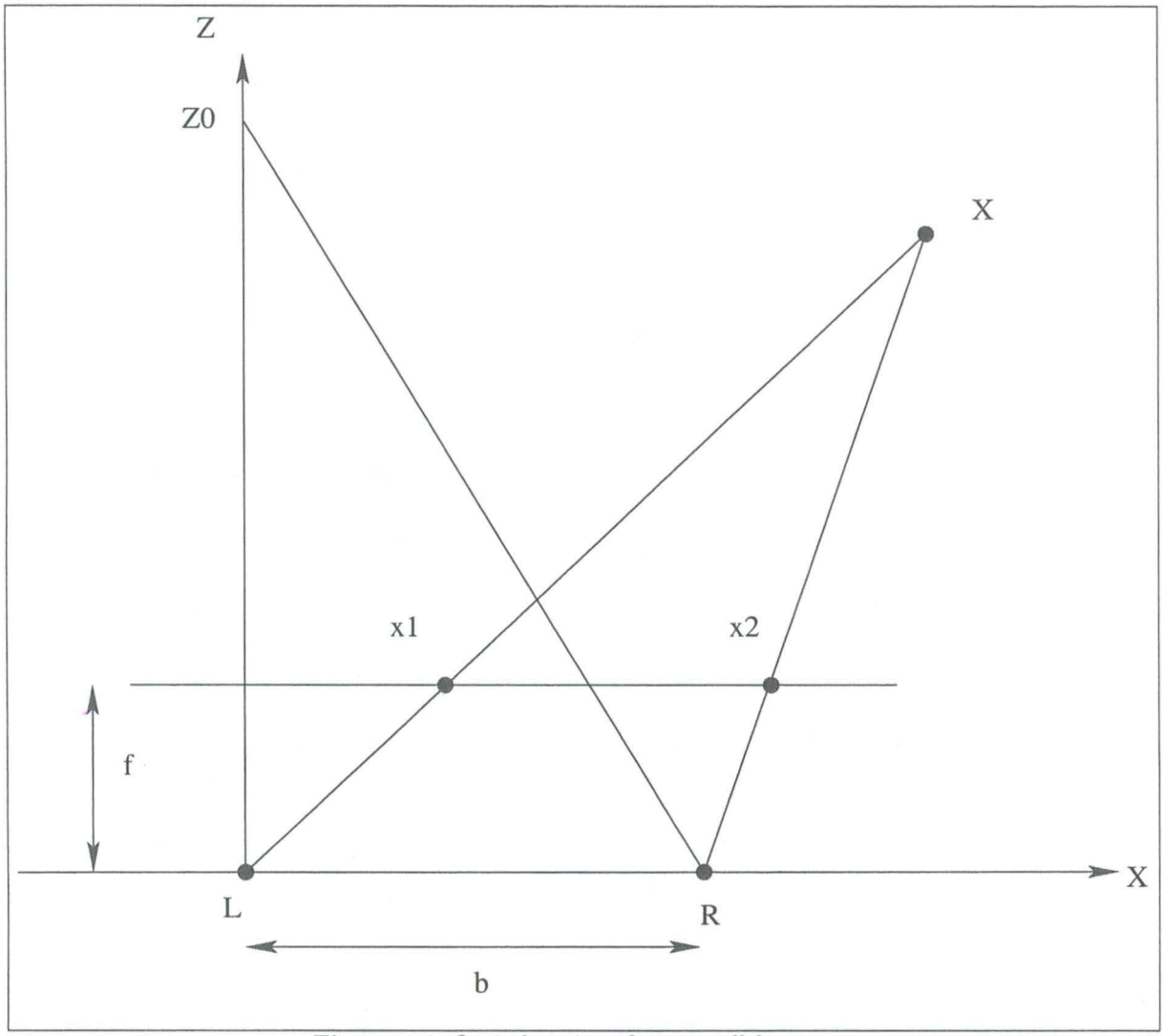


Figure 4.4: Optical setting of non-parallel cameras

If θ is the rotation angle, then $Z_0 = \frac{b}{\tan\theta}$. With this setting, the equations (4.9) are simplified and using once again the equations (4.5), the equations of stereo triangulation become

$$Z = \frac{(bf)}{(x_1 - x_2 + f\frac{b}{Z_0})}$$

$$X = x_1\frac{Z}{f}$$

Rotation around the X axis (ϕ)

Rotation around X axis only affects the Y coordinate. Let ϕ be the rotation angle, then stereo triangulation is

$$\begin{aligned} Z &= \frac{(bf)}{(x_1 - x_2)} \\ X &= x_1 \frac{Z}{f} \\ Y &= x_1 \frac{Z}{f} + \tan\phi Z \end{aligned}$$

Rotation around the Z axis (ψ)

Rotation around the optical axis is usually dealt with by rotating the image before applying matching and triangulation.

4.3.4 Mid-point technique

Unfortunately this geometric view is a little idealised. As summarised in [TV98], we have to take into account that camera parameters and correspondence locations in image space are known only approximatively. The rays will not actually intersect in 3D space. Their intersection can only be estimated as the point of minimum distance from both rays. This point will be located on a line segment orthogonal to both rays, as shown in figure 4.5. A standard approach first computes the end-points of this line segment. From these points, we can easily compute the mid-point X' that is the point in 3D space optimally close to the two non-intersecting rays.

- Let $\{\alpha x_1 \quad | \alpha \in \mathbb{R}\}$ be the ray, l, through C and x_1 (left origin and corresponding point).
- Let $\{\mathcal{T} + \beta\mathcal{R}^T x_2 \quad | \beta \in \mathbb{R}\}$ be the ray , r, through C' and x_2 (right origin and corresponding point), expressed in the left reference frame.
- Let $\{\gamma(x_1 \times \mathcal{R}^T x_2) \quad | \gamma \in \mathbb{R}\}$ be a vector w orthogonal to both l and r.

The problem is reduced to determine the midpoint, X', of the segment parallel to w that joins l and r (Figure 4.5).

This is straightforward because the endpoints of the segment can be computed by solving the linear system of equations:

$$\alpha x_1 - (\mathcal{T} + \beta\mathcal{R}^T x_2) + \gamma(x_1 \times \mathcal{R}^T x_2) = 0 \tag{4.10}$$

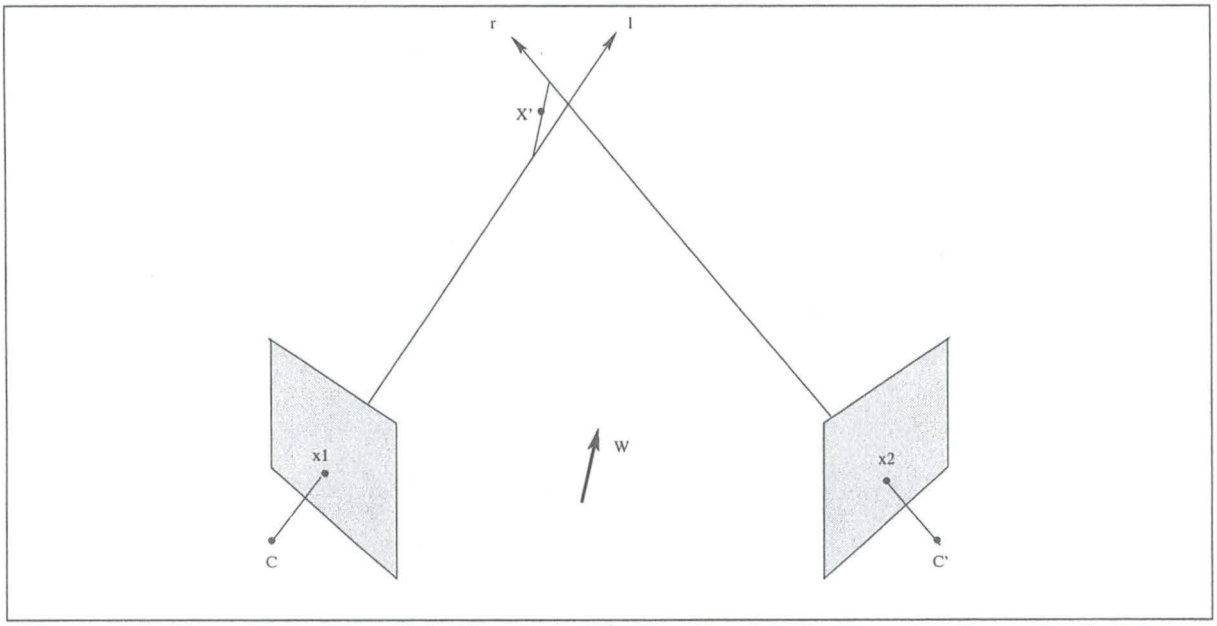


Figure 4.5: The approximation of the 3D point is X', the mid-point of a segment orthogonal to both rays.

Chapter 5

Edge detection

5.1 Edge detector

«In computer vision, edge detection is a process which attempts to capture the significant properties of objects in the image. These properties include discontinuities in the photometrical, geometrical and physical characteristics of object.»[ZT98]

Edges often occur at image locations of object boundaries, thus edge detection is used when we want to divide the image into areas representing different objects. Furthermore, representing the image by its edges has the great advantage that the volume of data is reduced significantly while retaining most of the useful image information. Since edges consist of mainly high frequencies, we can, in theory, detect edges by applying a high-pass frequency filter in the Fourier domain or by convolving the image with an appropriate kernel in the spatial domain. In practice, we perform edge detection in the spatial domain, first because it is computationally less expensive and because it often provides better results.

«Since image intensity is often proportional to scene radiance, physical edges are represented in the image by changes in the intensity function.»[ZT98]

We can highlight them by calculating the derivatives of the image.This is illustrated for the one-dimensional case in figure 5.1. We can see that the position of the edge can be estimated

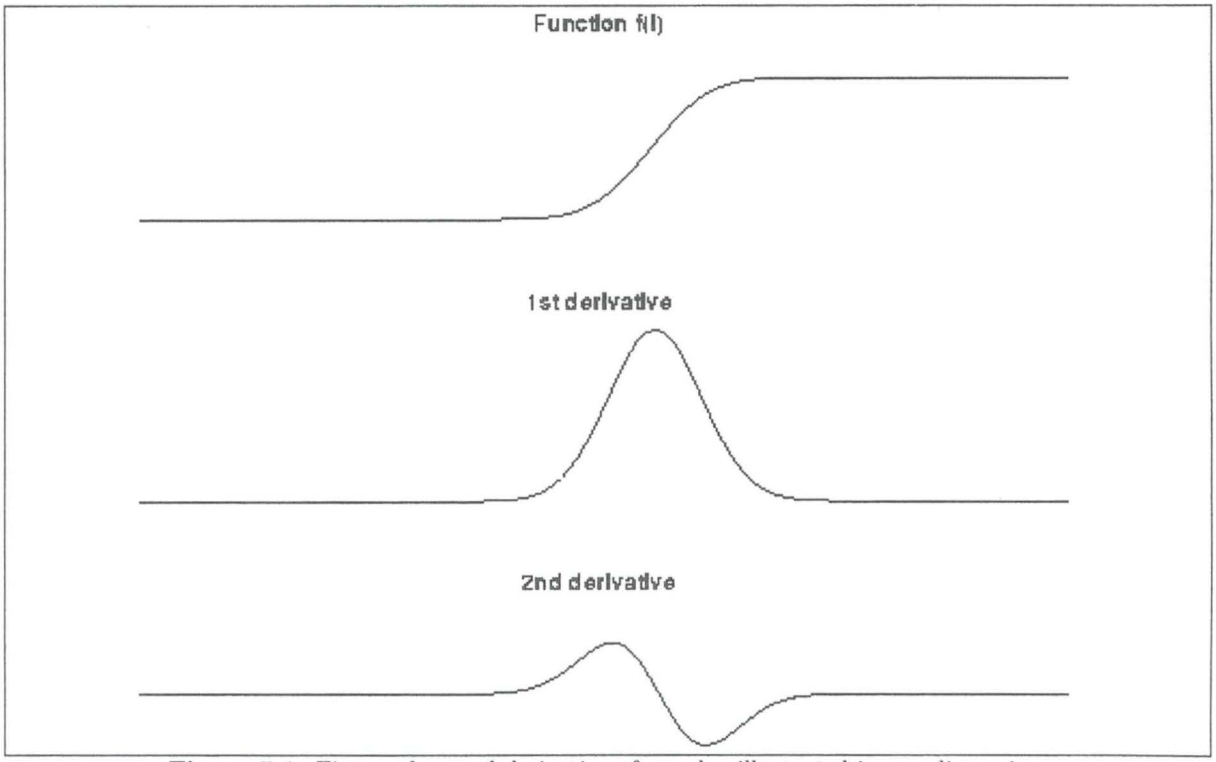


Figure 5.1: First and second derivative of an edge illustrated in one dimension.

with the maxima of the first derivative or with the zero-crossings of the second derivative,

one-dimensional function $f(x)$, the first derivative can be approximated by

$$\frac{d\,f(x)}{d(x)} = f(x+1) - f(x) \tag{5.1}$$

Different edge detection kernels which are based on the equation (5.1) enable us to calculate either the first and the second derivative of a two-dimensional image and then get the image intensity gradient as:

$$\nabla I(x,y) = \left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}\right) = (\nabla_x, \nabla_y) \tag{5.2}$$

There are two common approaches to estimate the first derivative in a two-dimensional image, *Prewitt compass edge detection* and *gradient edge detection.*

5.1.1 Convolution

Convolution is a mathematic operation which, from an image of dimension $(N \times M)$ and a kernel of dimension $(n_K \times m_K)$, attribute to each pixel a value which is a linear combination of the initial pixels.

If $I(x,y)$ is a pixel of an image and $K(i,j)$ is the kernel with $m_K \ll M$ and $n_K \ll N$, the resulting image $O(x,y)$ will be:

$$O(x,y) = \sum_{k=1}^{m_K} \sum_{l=1}^{n_K} I(x+l-1, y+k-1)K(k,l)$$

5.1.2 Noise

Real world signal is generally slightly different from the signal obtain by our production model. This gap is called *noise.* This noise comes from the process of signal capture and is not part of the ideal signal. Noise can usually be grouped in two classes:

- independent noise;
- noise dependent of the data of the image.

Independent noise can often be seen as an additional noise model, that means that the captured image $I(x,y)$ is the result of the sum of the *real* image $S(x,y)$ and a noise $n(x,y)$:

$$I(x,y) = S(x,y) + n(x,y)$$

The noise $n(i,j)$ has often a null average and is described by its variance σ_n^2. The impact of the noise can be described by the *Signal to Noise Ratio* (SNR) given by:

$$SNR = \frac{\sigma_S}{\sigma_n} = \sqrt{\frac{\sigma_I^2}{\sigma_n^2} - 1}$$

Every captured image is disturbed by a noise coming from the detector. Most of the time, this noise can be described by an independent and additional model where the noise $n(x, y)$ has a Gaussian distribution with null average and variance of σ. This implies that each pixel in the noised image is the sum of the real value of the pixel and a random value according to a distribution $\mathcal{N}(0, \sigma^2)$

The Gaussian noise can be reduced using a spatial filter, whose an example, among the most efficient, is the *Gaussian smoothing.*

5.1.3 Prewitt compass edge detector

This section and the following are based on [FPWW02], the interested readers can see there for more details and examples. The operation usually outputs two images, one estimating the local edge gradient magnitude, and one estimating the edge orientation of the input image. The technique uses a set of 8 (in general) convolution kernels, each of them sensitive to a particular orientation. Using these kernels, the image is convolved and for each pixel, the local edge gradient magnitude is computed with the maximum response of all 8 kernels at this pixel location

$$|G| = max(|G_i| : i = 1 \text{ to } n) \tag{5.3}$$

where G_i is the response of the kernel i at the particular pixel position and n is the number of convolution kernels. The kernel that yields the maximum response will give the approximation of the local edge orientation.

In figure 5.2, we can see two examples of the set of the 8 templates used by Prewitt. The

-1	+1	+1
-1	-2	+1
-1	+1	+1

(a) 0°

+1	+1	+1
-1	-2	+1
-1	-1	+1

(b) 45°

Figure 5.2: The Prewitt compass edge detecting templates sensitive to edges at 0° and 45°.

whole set of 8 kernels is produced by taking one of the kernels and rotating its coefficient circularly. Each of the rotating kernels is sensitive to an edge orientation ranging from 0° to 315° by steps of 45°, where 0° corresponds to a vertical edge (remember that the gradient is perpendicular to the edge).

The maximum response $|G|$ for each pixel is the value of the corresponding pixel in the output magnitude image. The values for the output orientation image are between 1 and 8, depending on which of the 8 kernels produces the maximum response. The edge magnitude and orientation of a pixel is then determined by the template that matches the local area of the pixel the best.

The *compass edge detector* is an appropriate way to estimate the magnitude *and* the

y-direction, the *compass edge detection* obtains the orientation directly from the kernel with the maximum response. On the other hand, the compass operator needs (here) 8 convolutions for each pixel, whereas the gradient edge detector needs only 2, one kernel being sensitive to edges in the vertical direction and one to the horizontal direction as we will see in the next section.

5.1.4 Gradient edge detector

Sobel edge detector

The Sobel operator performs a 2D spatial gradient measurement of an image and so emphasises regions of high spatial frequency that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input gray-scale image. In theory at least, the operator consists of a pair of 3×3 convolution kernels as shown in figure 5.3. One kernel is simply the other rotated by 90°.

-1	0	1
-2	0	2
-1	0	1

(a) S_x

-1	-2	-1
0	0	0
1	2	1

(b) S_y

Figure 5.3: The *Sobel* convolution filter which can be used to approximate the image intensity gradient.

These kernels are designed to respond maximally to edges running horizontally and vertically relative to the pixel grid, one kernel for each of the two perpendicular orientations.

The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each direction (call these ∇_x and ∇_y). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of the gradient.

The gradient magnitude is given by

$$\|\nabla I(x,y)\| = \sqrt{\left(\partial I(x,y)/\partial x\right)^2 + \left(\partial I(x,y)/\partial y\right)^2} = \sqrt{(\nabla_x)^2 + (\nabla_y)^2} \tag{5.4}$$

Typically an approximation is computed using

$$\|\nabla I(x,y)\| \simeq |\nabla_x| + |\nabla_y| \tag{5.5}$$

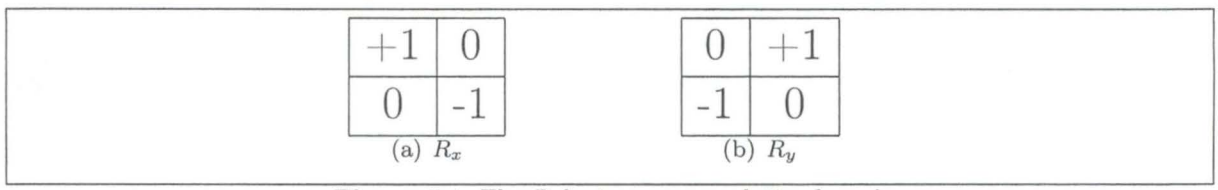
that is faster to compute.

The angle of orientation of the gradient is given by

Notice that if $\nabla_x = 0$ then the orientation of the gradient is vertical. Following these definitions, the orientation $\theta = 0$ is taken to mean that the direction of maximum contrast from black to white runs from left to right in the image, and other angles are measured anti-clockwise from this.

Roberts cross edge detector

The *Roberts cross edge detector* is very similar to the *Sobel edge detector*, the difference is the size of the convolution kernels. Here it is a pair of 2×2 kernels.



(a) R_x (b) R_y

Figure 5.4: The *Roberts* cross convolution kernel.

The main advantage of this technique is that is very quick to compute, only four input pixels are necessary to compute the value of each output pixel. But the disadvantage is that since it uses such a small kernel, it is very sensitive to noise.

Canny edge detection

Here is an overview of the Canny operator which was defined and proved in [Can86].

Canny introduced his edge detection technique in his thesis and proved that it is optimal according to the following criteria:

1. **to maximise the signal to noise ratio (SNR);**
2. **to minimise the distance between the response and the real edge[1];**
3. **to minimise the response to an unique edge.**

It takes as an input a gray-scale image, and produces as output an image showing the position of intensity discontinuities. The algorithm presented by Canny is made of four steps:

- **Step one** - Apply a Gaussian smoothing to the image, $G \otimes I$;
- **Step two** - Gradient calculation of $G \otimes I$, its magnitude and its direction;
- **Step three** - Non-maxima suppression of $\nabla(G \otimes I)$;
- **Step four** - Thresholding by hysteresis.

Step one - Gaussian smoothing. The image data is smoothed by a two dimensional Gaussian function of width defined as a user parameter. Actually the Gaussian smoothing is a convolution where the kernel $G(x, y)$, symmetric, with dimension $(n_K \times n_K)$ (where n_K is odd), is defined by

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu)^2+y-\mu^2}{2\sigma^2}}$$

which is the joint distribution function of two independent variables X, Y of identical distribution $\mathcal{N}(\mu, \sigma^2)$ where the average is defined by

$$\mu = \frac{(n_K + 1)}{2}$$

for n_K fixed.

As the variables X and Y are independent the convolution can be separated in two passes. One pass according to X (aligned with the $x-$axis), followed by one for Y (aligned with the $y-$axis). A point with coordinates (x, y) in the image I will have, after convolution, the value

$$O(x, y) = \sum_{k=1}^{n_K} I_k^{xy} G_X^T G_Y(k) = G \otimes I^{xy}$$

where I^{xy} is the image under-matrix of dimension $(n_K \times n_K)$ centred in (x, y) and I_k^{xy} is the k^{th} line of the matrix I^{xy}.

Notice that the interest in the separation of the convolution is that the algorithmic complexity is in the order of n_K^2 with the operator G and only n_K for G_X and G_Y.

Finally, to keep the same bounds for $O(x, y)$ than for $I(x, y)$, we have to normalise the kernel such that

$$\sum_{x=1}^{n_K} \sum_{y=1}^{n_K} G(x, y) = 1$$

Step two - Gradient calculation. Assuming two dimensional convolution at step one, the smoothed image data are differentiated with respect to the x and y directions. It is possible to compute the gradient of the smoothed surface of the convolved image function in any direction from the known gradient in any two directions.

The gradient of an image I at the point of coordinates (x, y) will be approximated by a convolution of kernel $(-1, 0, 1)$ according to X, noted $\nabla_x I^{xy}$, and of kernel $(-1, 0, 1)^T$ according to Y, noted $\nabla_y I^{xy}$.

Magnitude. Again, the magnitude of the gradient at point of coordinates (x, y) will be computed with

$$\sqrt{\nabla_x^2 I^{xy} + \nabla_y^2 I^{xy}}$$

but the value is generally approximated by

Direction. The direction of the approximated gradient $n = \nabla I / \|\nabla I\|$ will have an angle given by

$$\theta_n(x, y) = \arctan \frac{\nabla_y I^{xy}}{\nabla_x I^{xy}}$$

Step three - non-maxima suppression. Having found the rate of intensity change at each point in the image, edges must be placed now at the points of maxima, or rather non-maxima must be suppressed. We wish to suppress non-maxima perpendicular to the edge direction, rather than parallel to (along) the edge direction, since we expect a continuity of edge strength along an extended contour.

The non-maxima suppression will mark as edge point all the points which, in a 1-pixel neighbourhood, have a maximal magnitude in the direction of the gradient.

Thus, each pixel in turn become the centre of a 9-pixels neighbourhood, if the magnitude of this pixel is maximal in the direction of the gradient, it is accepted otherwise it is rejected (see figure 5.5).

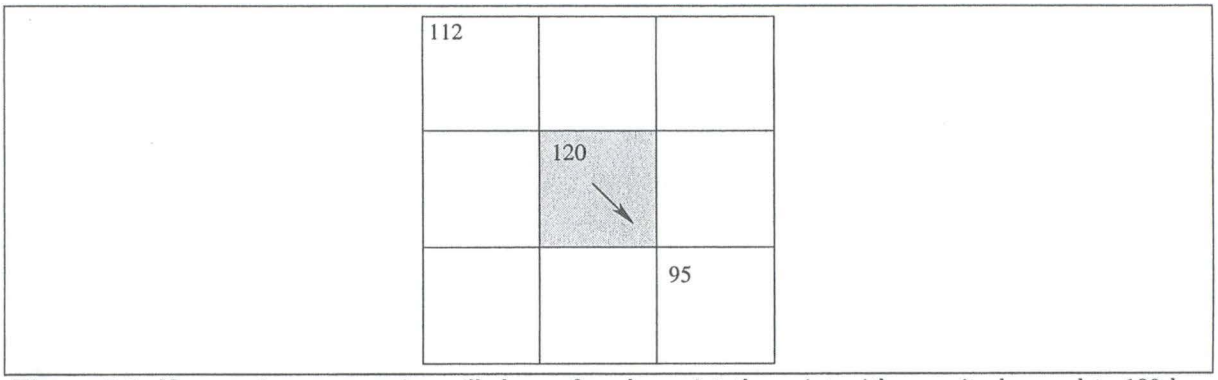


Figure 5.5: Non-maxima suppression will choose for edge point the point with magnitude equal to 120 because, in the direction of its gradient (represented by the arrow), the points in a 1-pixel neighbourhood have a lower magnitude (112 and 95)

Step four - Thresholding. In spite of the smoothing performed as the first step, the non-maxima suppressed magnitude image will contain many false edge fragments caused by noise, fine texture and shadows. For example a brutal changing in the image from white to gray or from white to black could be mistaken for an edge.

We should reduce these fragments that are useless information. One typical procedure is to apply a threshold to the non-maxima suppressed magnitude image. All values below the threshold are set to 0. After the application of this threshold, an array is obtained containing the edges detected in the input image.

However, in this method, it is very difficult to choose the correct value for the threshold and that involves trials and errors. Because of this difficulty, there may still be some false

The relation between the two threshold is often $t_2 \simeq 2t_1$, or $t_2 \simeq 3t_1$. With these threshold values, two thresholded edge images $T_1(x, y)$ and $T_2(x, y)$ are produced. If a value lies above the upper limit, it is automatically accepted and placed in the edge image $T_2(x, y)$. If a value lies below the low threshold it is automatically rejected. And if it lies between the two limits, the point is placed in $T_1(x, y)$.

The image T_2 has gaps in the contours but contains fewer false edges. With the double thresholding algorithm, the edges in T_2 are linked into contour. When it reaches the end of a contour, algorithm looks in T_1 at the locations of the 8 neighbours for edges that can be linked to the contour. This algorithm continues until the gap has been bridged to an edge in T_2. Thus, once we start to draw an edge, we don't stop before the magnitude of the gradient decreases considerably.

Note that the choice of the two thresholds and Gaussian deviation still remain the main factor of a correct edge detection as illustrated below.

Example We see with the figure 5.6 that the choice of the parameters is very important and could be different following the scene we are dealing with. The example of figure 5.6 is difficult because of the amount of details, thus different natural scenes (more or less complicated) will need different parameters values. Ideally, we should find a standard set of values that will be a compromise and produce acceptable[2] results for each scene, but it is not always possible.

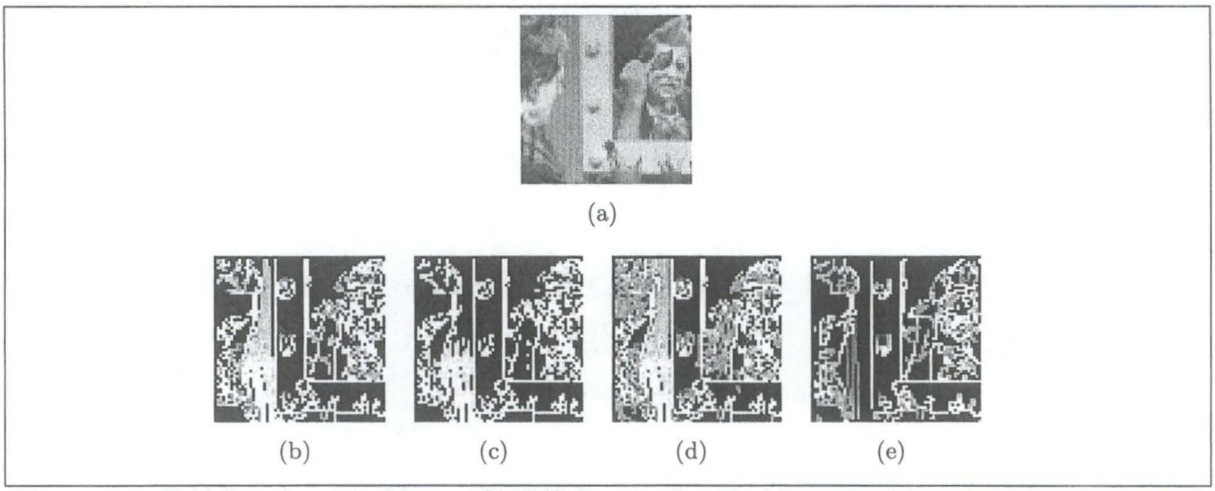


Figure 5.6: (5.6(a)) is the original image. (5.6(b)) Gaussian kernel with standard deviation 1.0 and upper and lower threshold of 255 and 1. (5.6(c)) Same kernel and upper threshold but with lower threshold of 220. (5.6(d)) Standard deviation = 1, lower threshold = 1 and upper = 128. (5.6(e)) Same threshold as (5.6(d)) but with Gaussian deviation = 2.

Part II

Details of the algorithm

Chapter 6

Panoramic generation

Introduction

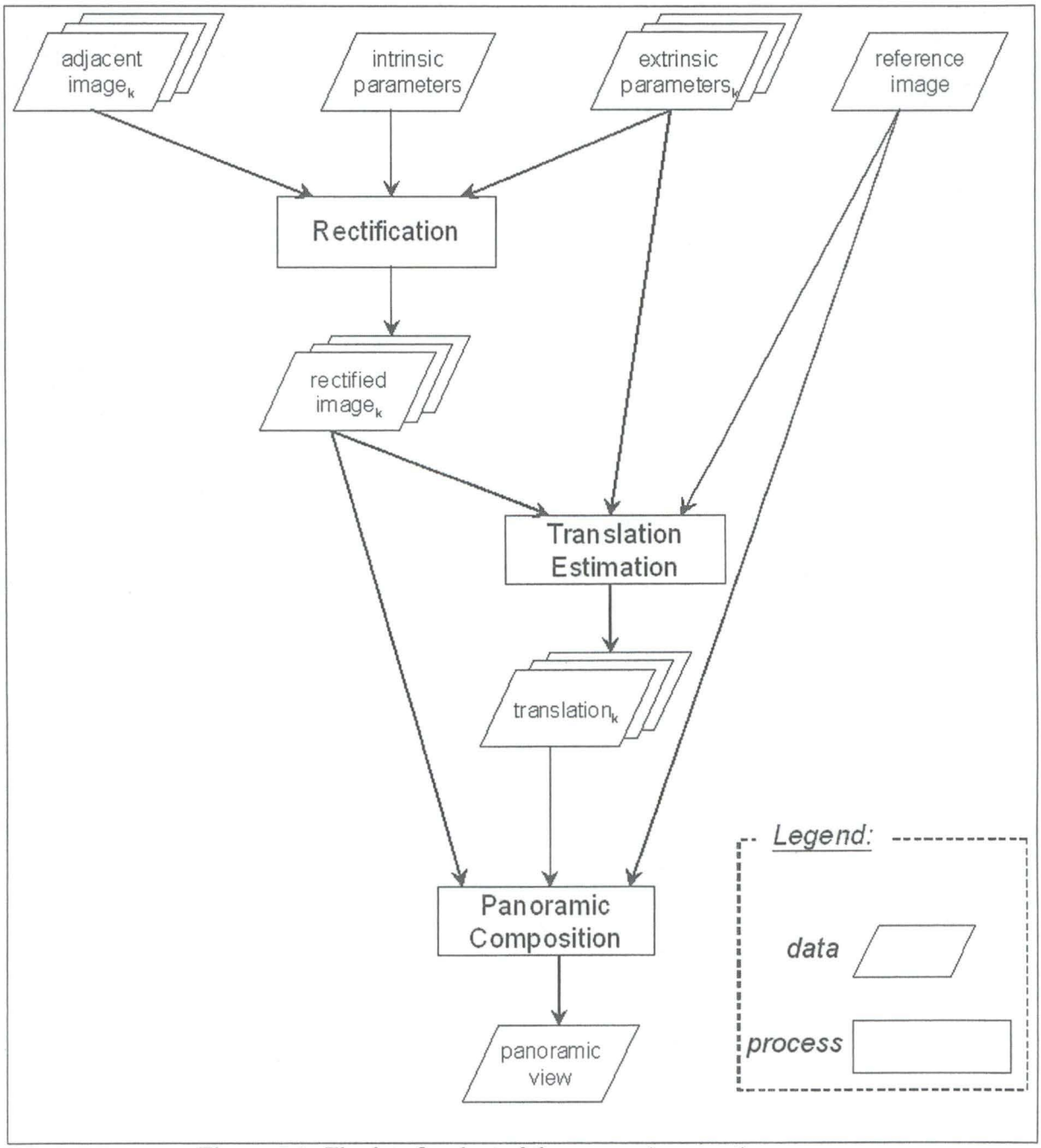


Figure 6.1: The data flowchart of the panoramic generation process.

To generate the panoramic image, none of the registration methods presented in chapter 3 is used. Indeed, the geometric transformations between source images are difficult to automatically estimate because of the very different viewpoints. We could use a hybrid

sually satisfactory result and not an accurate correspondence between the points in the images.

At a given time, the process computes a panoramic image from a set of received images in which we distinguish (see figure 6.2):

- the *reference image* coming from a camera which is considered as the origin of the coordinates system from the panoramic image point of view. This image remains unchanged all along the process;
- the *adjacent images* coming from several cameras surrounding the first one.

In short we perform a global alignment using a global transformation between the reference image and each adjacent image.

The overall flow of the proposed panoramic image generation has three main stages as it is illustrated in figure 6.1.

The first stage consists of rectifying each adjacent image in such a way that it is projected onto a plane which is parallel to the image plane of the reference image. Thus, the alignment problem is reduced to a single translation along the x-axis or the y-axis according to the relative position of the cameras. As we will see later, the camera intrinsic parameters and the extrinsic parameters of every surrounding camera are required to project the images onto new planes.

Next, the translation between every rectified image and the reference image is sought in a search space restricted on the one hand by the first stage and on the other hand by the position of the adjacent image in comparison with the reference image (see figure 6.2).

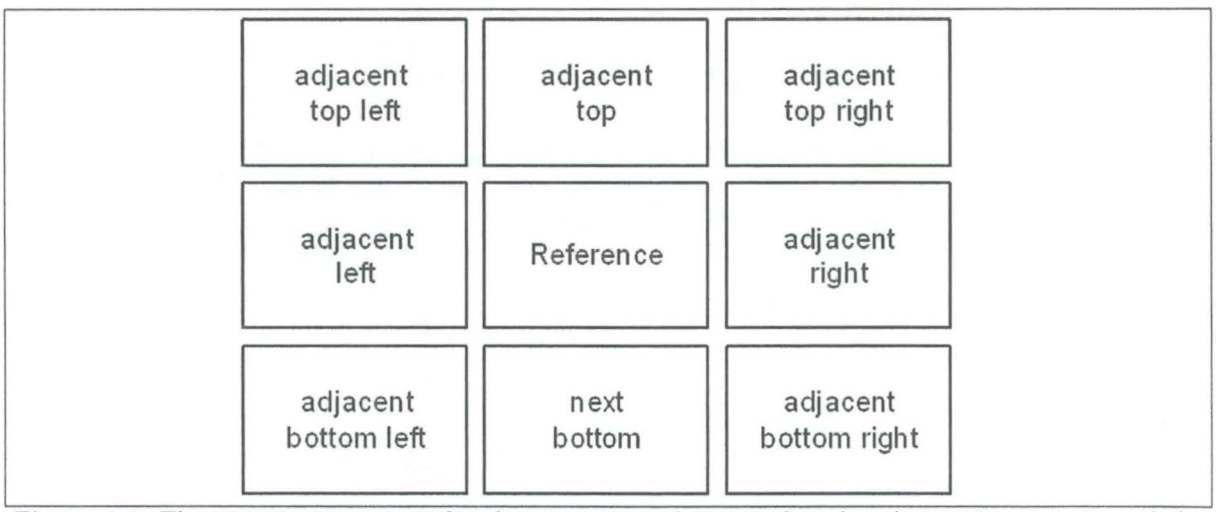
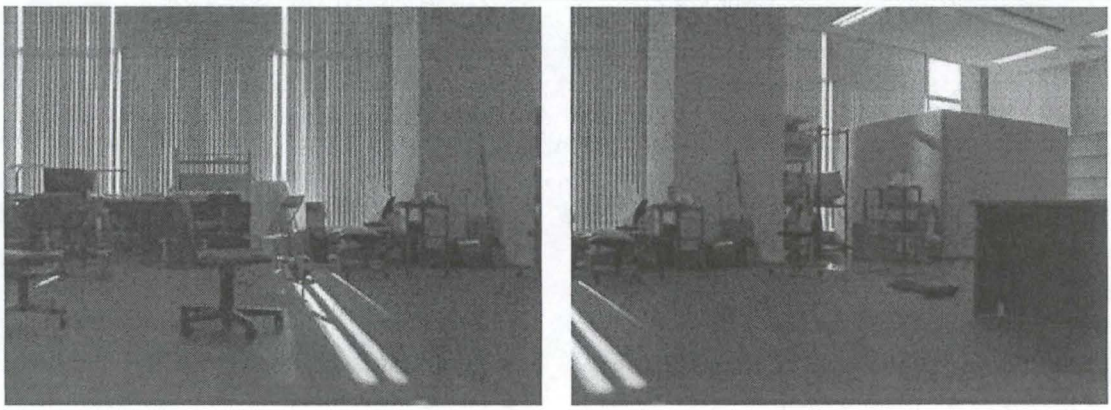


Figure 6.2: The cameras are supposed to be set up in such a way that the adjacent images surround the reference image. Thus, the search space is divided into 8 search areas according to the position of the adjacent images compared to the reference image.

reduce the misalignment errors between the overlaying parts of the images.

In what follows, each one of the stages is discussed in detail and then illustrated on the basis of two images shown in figure 6.3.



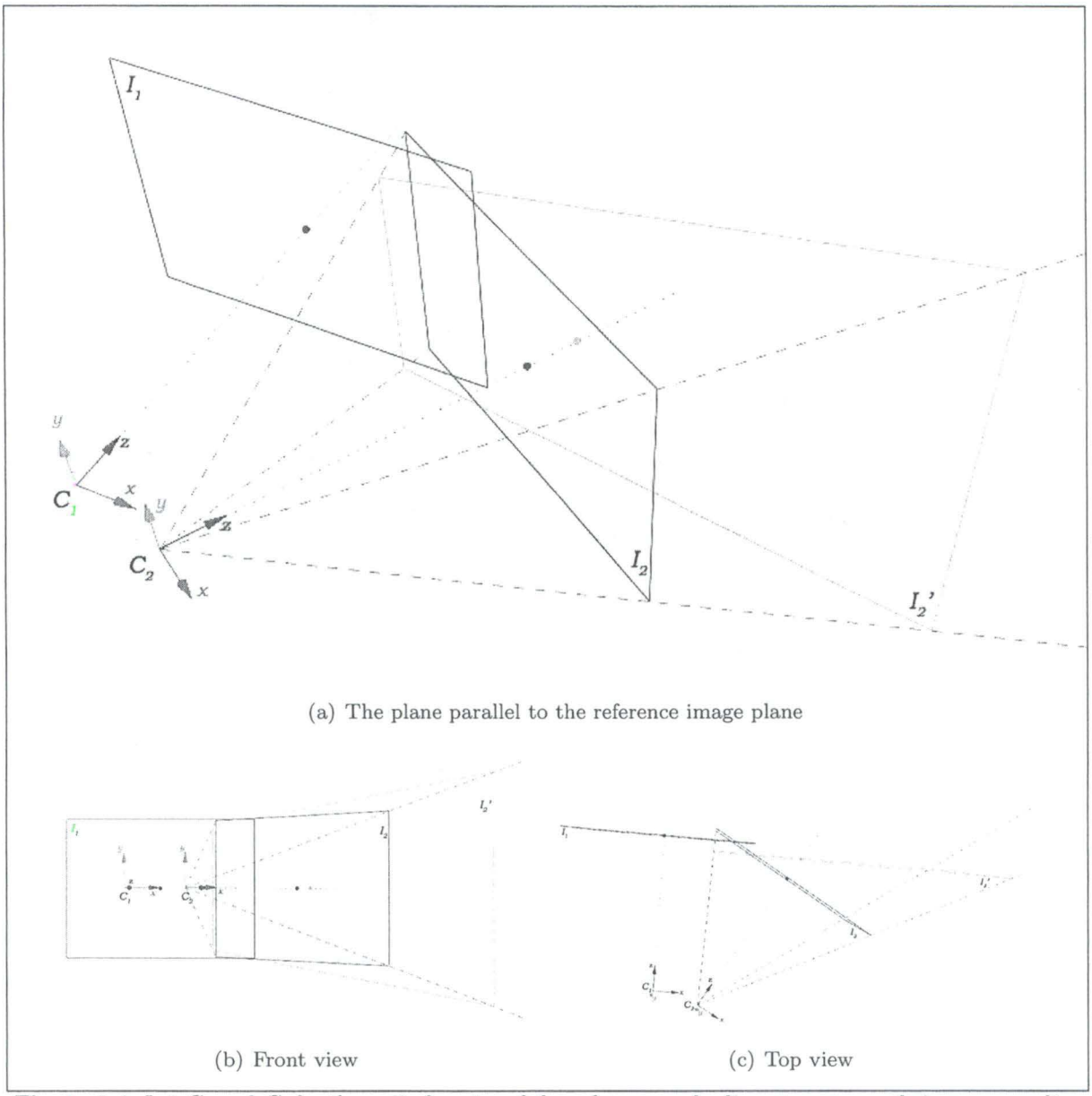
(a) The reference image (b) An adjacent image

Figure 6.3: Two images taken with a hand-held camera. The camera was moved of 7cm on the right and of 3cm behind between the two photographs but also turned behind of 30 ° towards the right. In what follows, we consider the image 6.3(a) as the reference image and the one on the right as the adjacent image 6.3(b)

6.1 The rectification

The objective of this stage is to reduce the geometric transformation between the images to a simple translation. For that, an adjacent image is rectified in a way similar to the method explained in section 2.4 "Rectification".

Instead of computing a rectification matrix and transforming each image, the proposed method transforms the adjacent images relative to the reference image. First we assume that the translation along the z-axis is small compared to the depth of the scene. Thus it can be ignored and we simply rotate the image plane of each adjacent image according to the rotation between the reference camera and the respective camera as illustrated on figure 6.4.



(a) The plane parallel to the reference image plane

(b) Front view

(c) Top view

Figure 6.4: Let C_1 and C_2 be the optical centre of the reference and adjacent cameras, their corresponding image planes are respectively I_1 and I_2. The rectification of the image on the plane I_2 consists of projecting this image onto the plane I_2' parallel to the plane I_1 of the reference image.

Given the angles ϕ, θ, ψ of rotation around the x, y and z axis between the reference camera and the considered camera, we compute the rotation matrix $\mathcal{R}'$ from the opposite values in such a way to obtain the rotation to apply to the image plane.

$$\mathcal{R}'_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos -\phi & \sin -\phi \\ 0 & -\sin -\phi & \cos -\phi \end{pmatrix} \tag{6.1}$$

$$\mathcal{R}'_y(\theta) = \begin{pmatrix} \cos -\theta & 0 & -\sin -\theta \\ 0 & 1 & 0 \\ \sin -\theta & 0 & \cos -\theta \end{pmatrix} \tag{6.2}$$

$$\mathcal{R}'_z(\psi) = \begin{pmatrix} \cos -\psi & \sin -\psi & 0 \\ -\sin -\psi & \cos -\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6.3}$$

$$\mathcal{R}' = \mathcal{R}'_x \mathcal{R}'_y \mathcal{R}'_z \tag{6.4}$$

As noticed in section 2.4, we implement the rectification backwards using the inverse matrix $\mathcal{R}'^{-1}$ to assign pixel values in the new image plane. For each pixel (x', y'), we obtain the coordinates (x, y) of the pixel in the original image as:

1. transform the coordinates from frame to image plane[1]

$$\hat{x} = (x - O_x)\ S_x \tag{6.5}$$

$$\hat{y} = (y - O_y)\ S_y \tag{6.6}$$

2. Compute

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathcal{R}'^{-1} \begin{bmatrix} \hat{x} \\ \hat{y} \\ f \end{bmatrix} \tag{6.7}$$

3. finally obtain the coordinates (x, y)

$$x = \frac{\left(u\ \frac{f}{w}\right)}{S_x} \tag{6.8}$$

$$y = \frac{\left(v\ \frac{f}{w}\right)}{S_y} \tag{6.9}$$

These coordinates of pixels in the original image are not integer, we use bilinear interpolation[2] to obtain the pixels value. On figure 6.5, we can see the result of the rectification.

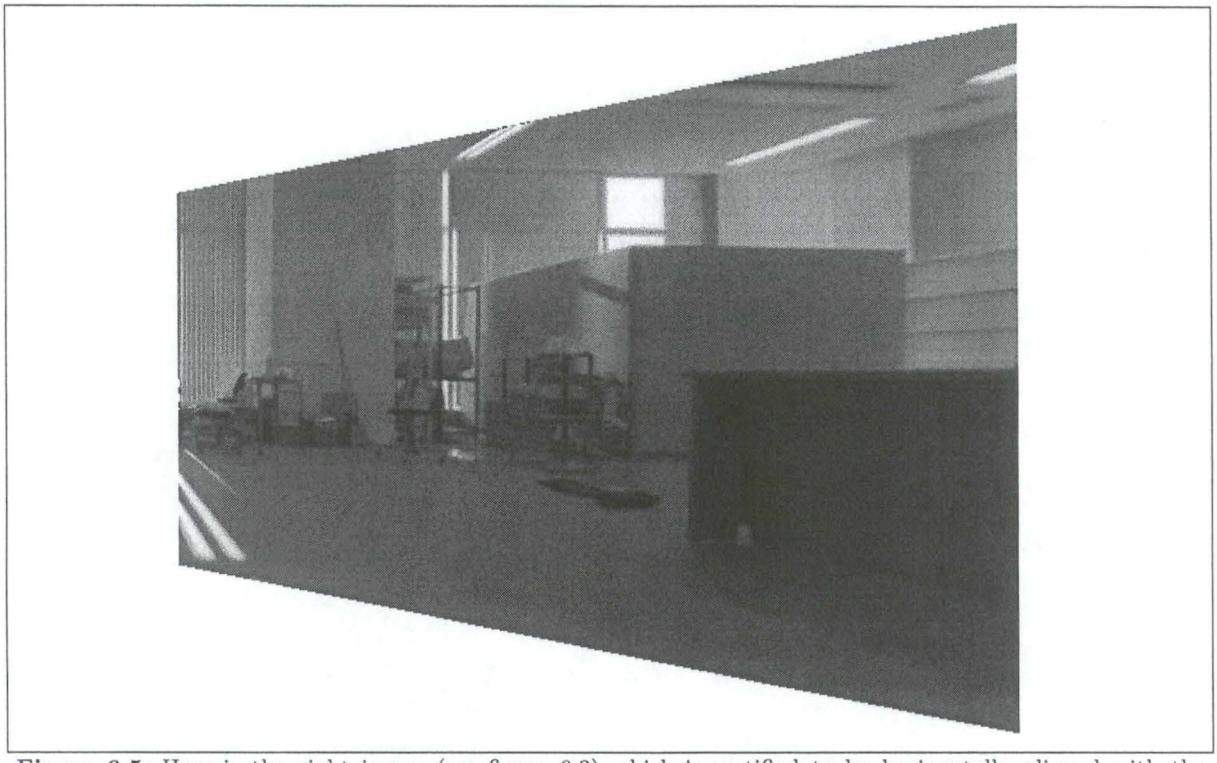


Figure 6.5: Here is the right image (see figure 6.3) which is rectified to be horizontally aligned with the reference image.

6.2 The estimation of the translation

Given the overlapping images, we have the reference image[3] and we want to estimate each one of the translations $\overline{\mathcal{T}}(\Delta_x, \Delta_y)$ to be applied to each adjacent image so that the overlapping areas of the images are aligned.

Thanks to the location of the cameras, we can restrict the search space $\mathbb{T}$ of the possible translations according to the position of the adjacent image in comparison with the reference image. For example, let's the adjacent image be on the right, the space $\mathbb{T}$ of the possible translations is from $\mathcal{T}(-width, 0)$[4] to $\mathcal{T}(0, 0)$[5]. We can reduced this interval still further by estimating the maximum possible overlap between the two images.

With this end in view, a common approach is discussed below without considering the first stage of rectification. This algorithm is based on an exhaustive search using a coarse grid proposed in [ZB01]. Afterwards an example is shown in the case of rectified images.

[3] Its origin (for example the upper-left corner of the image) is the origin of the coordinates system in which we are working.

[4] In the case of $\mathcal{T}(-width, 0)$ (where *width* is the width of the image), only the left edge of the adjacent

6.2.1 The exhaustive search method

To improve the reliability of the estimation, all the available information is exploited i.e. all the pixels in the overlapping area are considered.

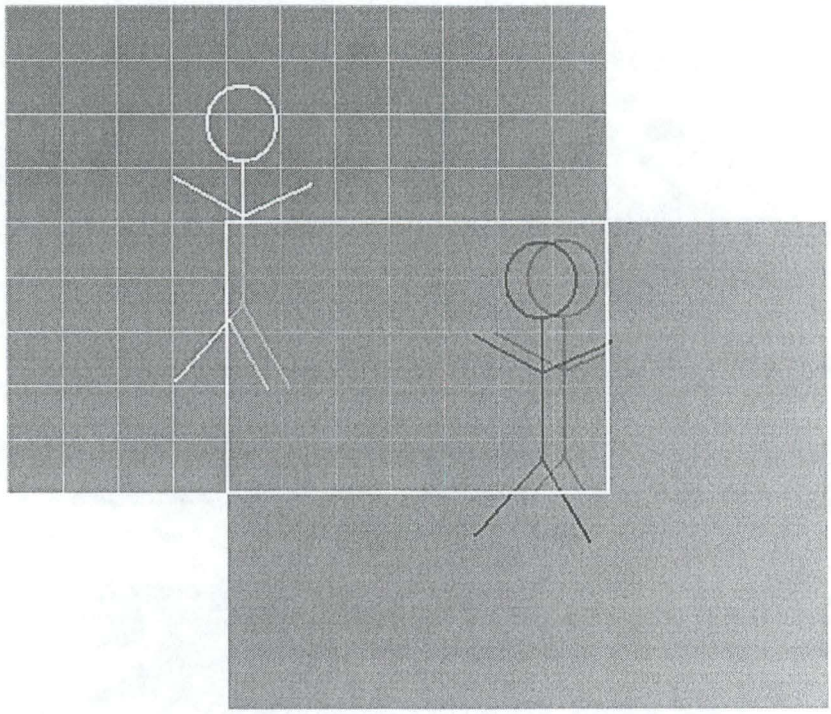
The algorithm iteratively computes a similarity measure[6] S (6.10) at all possible translations $\mathcal{T}(\Delta_x, \Delta_y)$ to exhaustively search the space $\mathbb{T}$ of possible translations and the one which maximises the measure S is estimated to be the best translation $\overline{\mathcal{T}}(\Delta_x, \Delta_y)$. Thus, the similarity measure S expresses the accuracy of the matching in progress and considers the intensity value of all the pixels in the current overlapping area $\mathcal{C}$:

$$S = \frac{2 \sum_{(x,y)\in\mathcal{C}} I_1(x,y) I_2(x+\Delta_x, y+\Delta_y)}{\sum_{(x,y)\in\mathcal{C}} \left(I_1(x,y)^2 + I_2(x+\Delta_x, y+\Delta_y)^2 \right)} \tag{6.10}$$

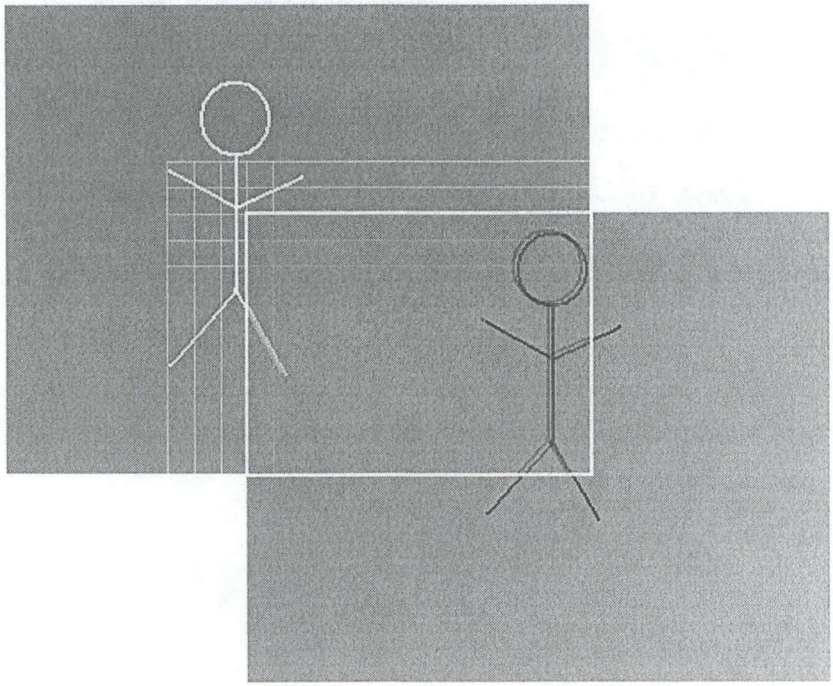
As in [ZB01], a coarse grid is applied between the two images to find a first estimation of the translation i.e. a large step value between each possible translation $\mathcal{T}(\Delta_x, \Delta_y)$ is used. Once the best translation $\hat{\mathcal{T}}_1$ is found for the first step, a new search is performed on a finer grid in the vicinity of $\hat{\mathcal{T}}_1$, using a smaller searching step (see figure 6.6). The vicinity of $\hat{\mathcal{T}}_1$ is defined as the search space from the translation estimated before $\hat{\mathcal{T}}_1$ to the translation considered after $\hat{\mathcal{T}}_1$, i.e. the space centred on $\hat{\mathcal{T}}_1$ and of size $2 \times step$.

This is repeated until an accuracy of 1-pixel is reached and finally obtain the estimation $\overline{\mathcal{T}}$. In this way, the algorithm is robust and is not so computationally expensive because of taking all the pixels in the overlapping area into consideration and of the exhaustive search. Note that the value used for the first step must be quite selected in order to obtain good results:

- a low value increases the computation time;
- a large value increases the probability of false matching.



(a) The first step



(b) The second step

Figure 6.6: In the first picture 6.6(a), the coarse grid is drawn on the reference image with a first step of 32 pixels. The pixels which are considered in the computation of the best similarity measure are surrounded in white, this yield to the first estimation $\hat{\mathcal{T}}_1$. On the next stage 6.6(b), a smaller

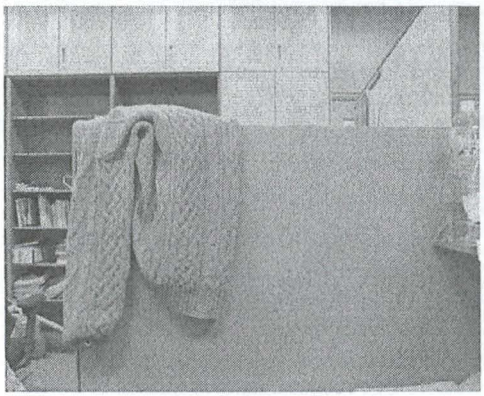
On figure 6.7 and 6.8, we can see some examples obtained using the algorithm described above. In the mosaic view, the adjacent image simply overlays the reference image according to the estimated translation.

(a) The first image

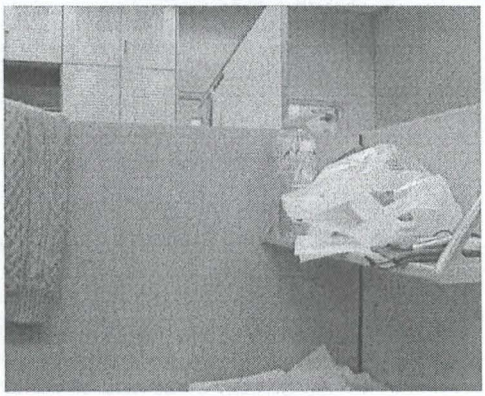
(b) The second image above on the right

(c) The images overlayed with the estimated translation

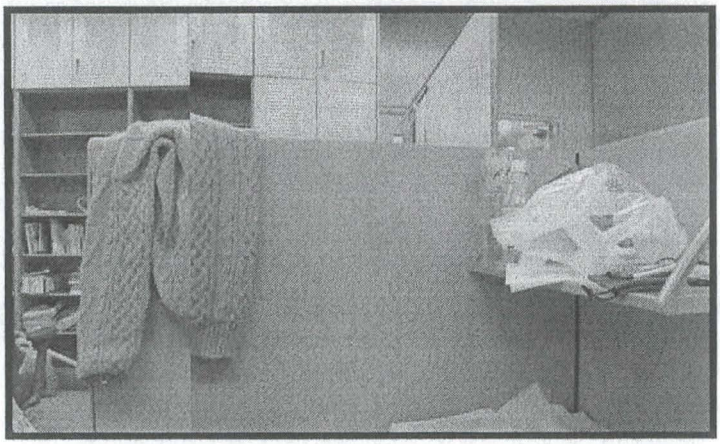
Figure 6.7: The input images 6.7(a) and 6.7(b) come from the same picture and the image 6.7(b) has been distorted in order to test the robustness of the algorithm.



(a) The right image



(b) The left image

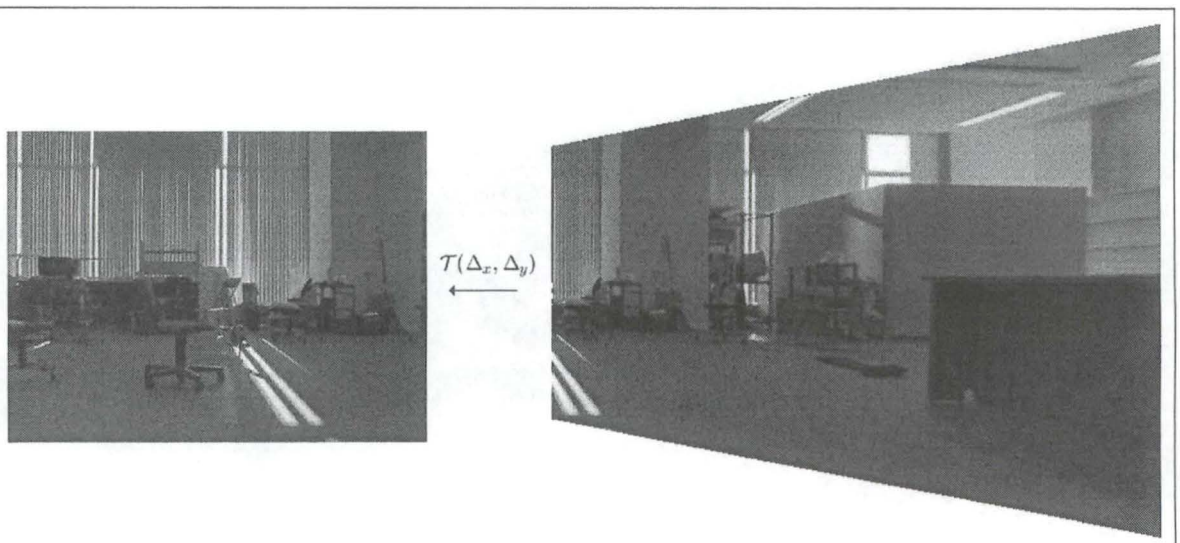


(c) The images overlayed with the estimated translation

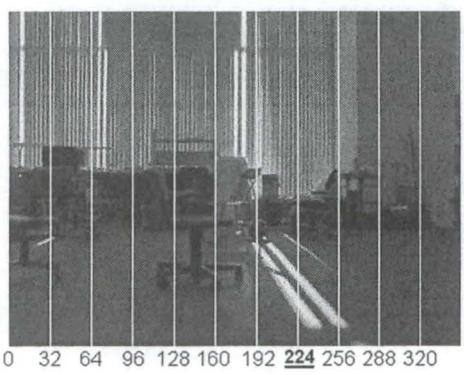
Figure 6.8: An example with images 6.8(a) and 6.8(b) taken with a hand-held camera. Despite the small motion (rotation and translation) between the two camera poses and the change in illumination, the translation is estimated well and the images are approximately aligned 6.8(c). Note that the search space $\mathbb{T}$ was limited to horizontal translation along the x-axis.

6.2.2 In our implementation

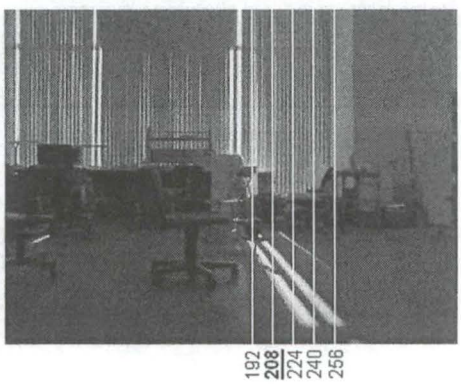
Given the reference image and the rectified image, the coarse grid method is applied to find the first estimation of the translation (see figure 6.9).



(a) The reference image and the rectified adjacent image



(b) The first step (reference image)



(c) The second step (reference image)

Figure 6.9: On the picture 6.9(b), the coarse grid is drawn on the reference image with a step of 32 pixels. The similarity measure (see table 6.1) has a maximum value for $\Delta_x = -224$ i.e. when the overlapping area in the reference image is from the 224^{th} to the 352^{th} pixel along the x-axis. On the picture 6.9(c), the coarse grid in the vicinity of $\hat{\mathcal{T}}_1$ is drawn in the reference image. The next step is of 16 pixels in the vicinity of 224^{th} pixels along the x-axis (see table 6.2).

Δ_x	value of S
0	0.597674644100379
-32	0.593988983561173
-64	0.6105110949544069
-96	0.6041678857347818
-128	0.6041678857347818
-160	0.6268926529702445
-192	0.7483860784409733
-224	**0.7990459416888565**
-256	0.698801578437786
-288	0.5026258378241202
-320	0.5026258378241202

Table 6.1: The values of the similarity measure S (equation 6.10) obtained during the first step of 32 pixels. The first column is the translation along the x-axis between the reference image and the adjacent image which is on the right. Note that the overlapping area in the reference image starts at pixel $-\Delta_x$ along the x-axis.

Δ_x	value of S
-192	0.7483860784409733
-208	**0.8346878527534598**
-224	0.7990459416888565
-240	0.7736619605441296
-256	0.698801578437786

Table 6.2: The values of the similarity measure S obtained during the second step of 16 pixels. The search space is from $\Delta_x = -192$ (the translation estimated before $\Delta_x = -224$) to $\Delta_x = -256$ (the translation estimated after $\hat{T}_1$). Here, the similarity measure is maximised with $\Delta_x = -208$.

6.3 The composition

Once we have rectified all the adjacent images and have found their respective translation $\overline{T}(\Delta_x, \Delta_y)$ related to the reference image, the last stage is to combine the sequence of images into a single panoramic view.

Simply aligning the images produces visible defects (i.e. misalignment errors) and brightness seams (i.e. the edges of the component images) in the mosaic between regions covered by several images because of changing in image brightness. These variations of brightness are mainly due to different illumination orientations and conditions. Consequently, methods are developed in order to create seamless mosaic from several images. This is the blending process which must smooth all the discontinuities in intensity and colour while preserving image sharpness.

Several ways are described below and they are presented in the ascending order of their

6.3.1 Average value

In the case where image alignment is close to perfect, it is desirable to use all the images in the overlapping area to produce the panoramic mosaic. So we can simply compute the pixels value in the mosaic by averaging the corresponding values in all overlapping images as shown on figure 6.10.

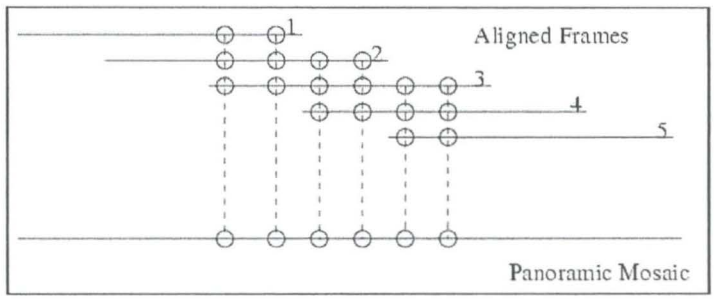


Figure 6.10: The principle of the average value. The pixel values in the panoramic mosaic are computed as the average of the pixel intensities in the aligned frames.

In addition to a simple mean, the median is often used to compute the pixel values in the mosaic from all overlapping regions. The example shown on figure 6.11 is the result obtained with the mean value.

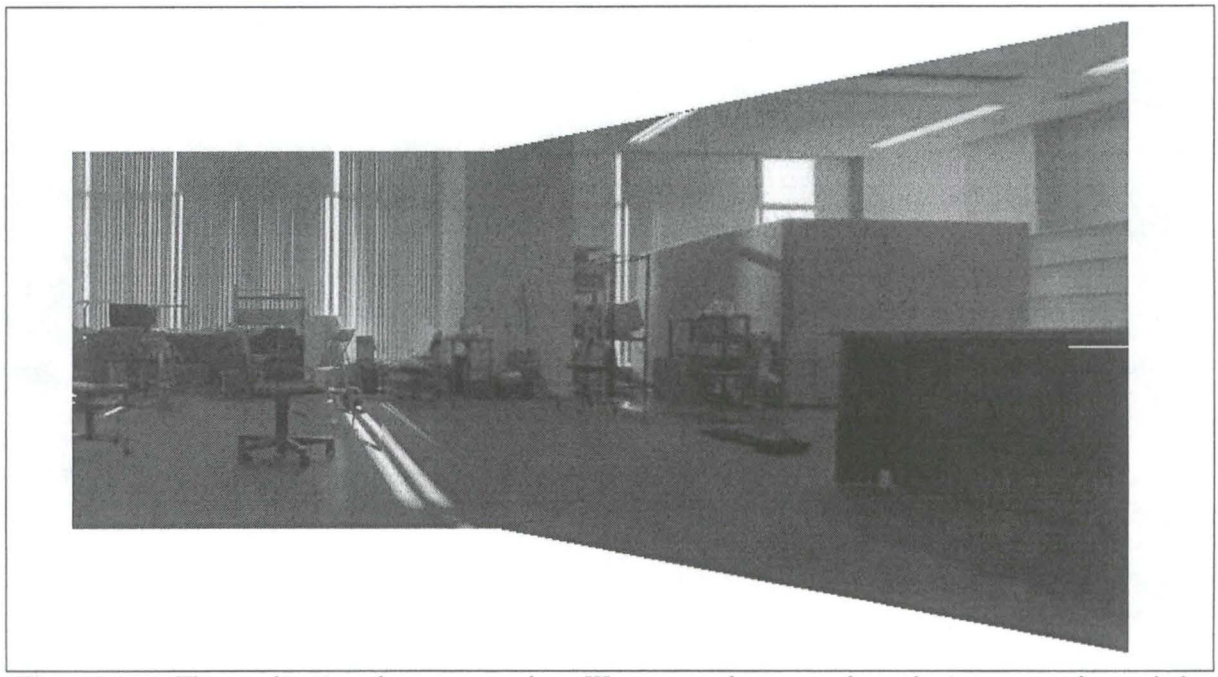


Figure 6.11: The result using the average value. We can see the seam where the images overlap and the overlapping area between the images is blurred. There is also deterioration of image quality due to the approximative alignment.

6.3.2 Single value

When the alignment between images is not perfect, averaging may result in blurring and in deterioration of image quality (see figure 6.11). In this case it is preferred to select only one of the input images to represent a region in the mosaic. Such a selection should be done to minimize effects of misalignment. The most logical selection is to select from each image that part closest to its centre (see figure 6.12). There are two reasons for that selection :

- alignment is usually better at the centre than at the edges of the pictures;
- image distortion is minimal at the centre of the image.

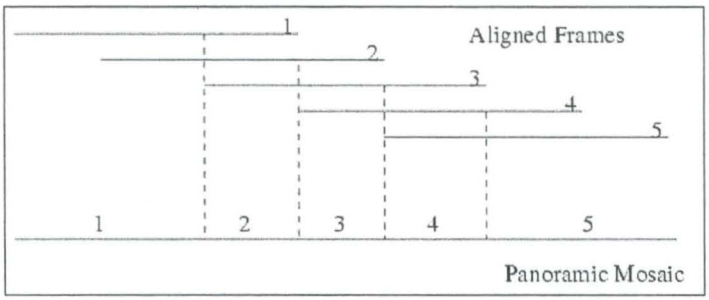


Figure 6.12: The principle of the single value. The pixel values in the panoramic mosaic are taken from a single frame whose centre, after alignment, is closest to the corresponding pixel. So the value of all the pixels in *Region 2* of the mosaic is therefore taken from the strip at the centre of *Image 2*, etc...

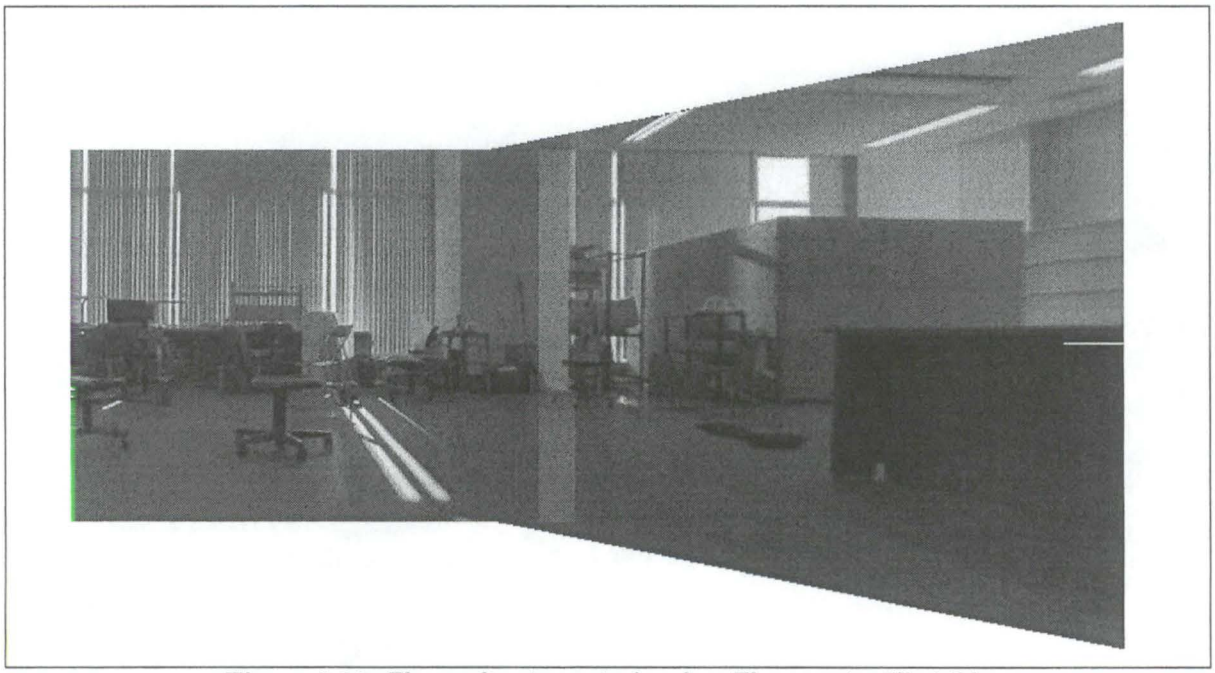


Figure 6.13: The result using a single value. The seam is still visible

6.3.3 Weighted average value

Due to different viewpoints, the images are not perfectly aligned despite the rectification and the estimation of the translations. Thus, the first two methods presented above provide unsatisfactory results. In order to hide the visible edges of the component images and to reduce the effects of misalignment errors, we use a weighted average value.

The objective is to gradually increase the contribution (in the final mosaic view) of pixels from each component image proportionally to their distance to the edge. We obtain the following formula to compute the value of the pixel (x, y) in the panoramic view:

$$Pan(x,y) = \frac{\sum_{k=1}^{M} \omega(x', y', k) I_k(x', y')}{\sum_{k=1}^{M} \omega(x', y', k)} \tag{6.11}$$

where $Pan()$ is the intensity function of the panoramic image, I_k is the intensity function of the k^{th} component image and M is the number of component images.

Two weight functions $\omega(x', y', k)$ are commonly used:

- weight with *triangle function*;
- weight with *distance*.

The weight with triangle function

In [Sze96], the weighting function $\omega()$ is a simple bilinear function:

$$\omega(x,y) = \omega_t(x)\omega_t(y) \tag{6.12}$$

where $\omega_t()$ is a triangle (hat) function that is closer to 1 as the coordinates (x, y) are nearer to the centre of the image. In practice we have the triangle function $\omega(x, y, k)$ as follows:

$$\omega(x,y,k) = \left(1 - \left|\frac{x}{width_k} - \frac{1}{2}\right|\right) \left(1 - \left|\frac{x}{height_k} - \frac{1}{2}\right|\right) \tag{6.13}$$

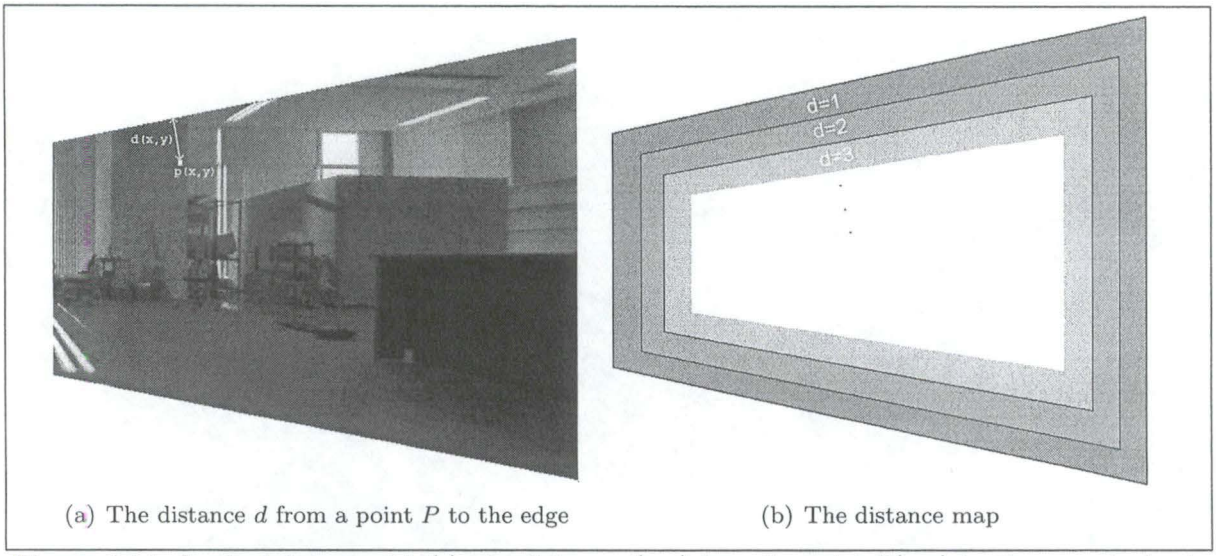
where $width_k$ and $height_k$ are the width and height of the k^{th} image respectively.

The weight with distance

In [SS97], a distance map $d(x, y)$ is first computed (see figure 6.14). We can either use the *Euclidean* distance[7] or the *city block* distance [8] to the nearest edge to build the map.

[7]The *Euclidean* distance is the length of the straight line between two points $P(x, y)$ and $Q(u, v)$. It is thus computed as $d_e = \sqrt{(x-u)^2 + (y-v)^2}$ in such a way that all pixels are at equal distance d_e and form a circle.

[8]For two given points $P(x, y)$ and $Q(u, v)$, the *city block* distance measures the path between the points based on a 4-connected neighbourhood and is computed as $d_{cb} = |x-u| + |y-v|$. Consequently, pixels whose



(a) The distance d from a point P to the edge (b) The distance map

Figure 6.14: On the left image 6.14(a), the distance $d(x, y)$ from the point $P(x, y)$ to the nearest edge is drawn. On the right image 6.14(b), the distance map i.e. an image where the distance to the edges has been pre-computed in all locations.

Back to the function $Pan(x, y)$, the weight function $\omega()$ is simply a monotonic function and $\omega(d(x, y)) = d(x, y)$ is currently used, where $d(x, y)$ gives the distance from the pixel $P(x, y)$ to the nearest edge as shown on figure 6.15.

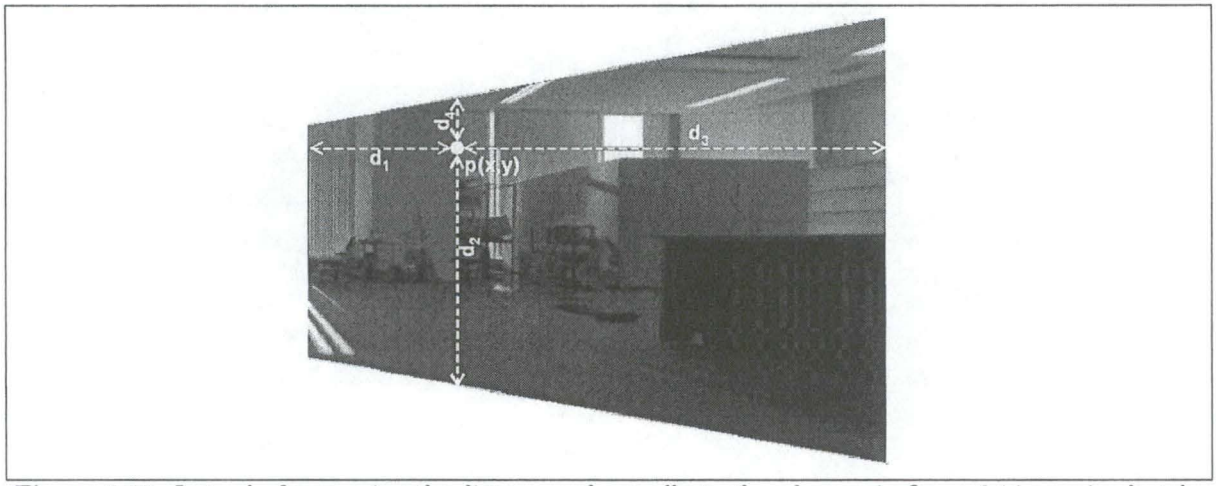
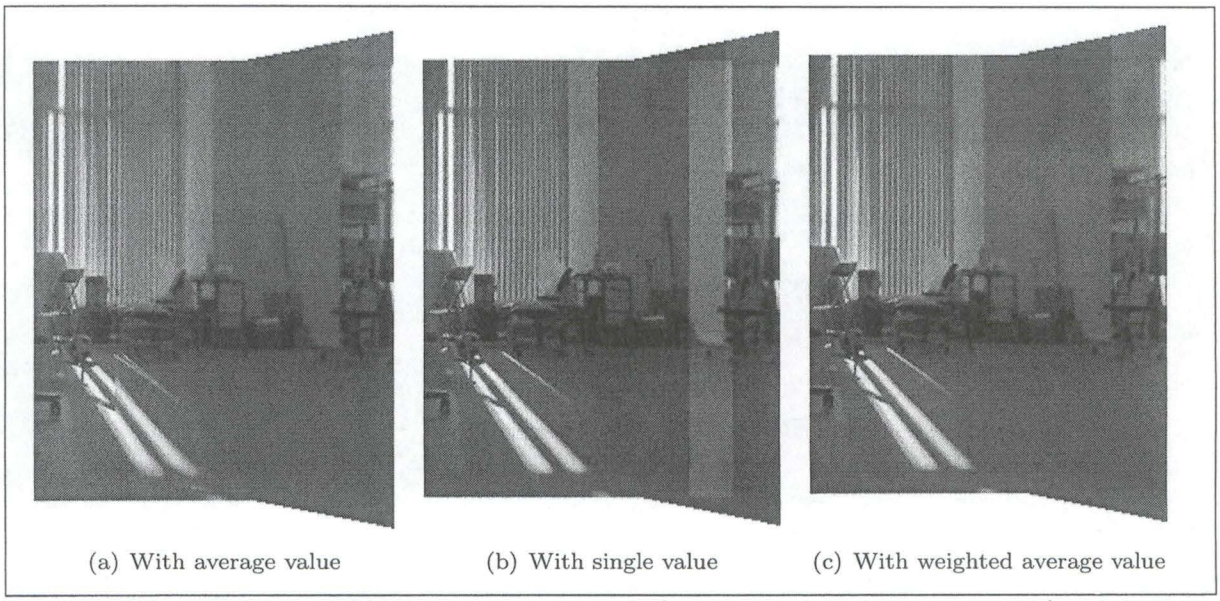


Figure 6.15: Instead of measuring the distance orthogonally to the edges as in figure 6.14, we simply take the distances d_1,d_2,d_3 and d_4 that follow the column x and the row y of the coordinates (x, y) to reduce the complexity of the measurement. Thus, the distance $d(x, y)$ to the nearest edge is chosen amongst the quadruplet (d_1, d_2, d_3, d_4) and gives d_4.

As [WYZ01] underlines it, the weight with distance is more efficient than the weight with triangle function. By using the distance map, the contribution of pixels at the edges is null and gradually increases towards the centre of the component image. With a triangle function, the pixels at the edges contribute 50% to the mosaic image, resulting in a slightly visible seam.

6.3.4 Comparison of the various approaches



(a) With average value (b) With single value (c) With weighted average value

Figure 6.16: On picture 6.16(a), the misalignment errors (e.g. the rays of light on the ground) and the seam between the two images are visible. In addition, the region where the images overlap is blurred (e.g. the curtains and the rack in the middle of the picture). Using a single value 6.16(b) hides the misalignment errors and the blurring effect in the overlapping area but there is a seam that is still visible (i.e. the gray stripe on the right). At least, using the average value weighted with distance disguises all these artifacts well as shown in the picture 6.16(c).

6.3.5 In our implementation

As discussed earlier, the best result is obtained with the weight with distance. So we compute the distance to the nearest edge as in figure 6.15 and the pixel weight is then proportional to this distance to some power n with the function $\omega(x) = x^n$. In practice we have:

$$Pan(x,y) = \frac{\sum_{k=1}^{M} \left(d_k(x',y')\right)^n I_k(x',y')}{\sum_{k=1}^{M} \left(d_k(x',y')\right)^n} \tag{6.14}$$

where M is the number of overlapping images, $d_k(x',y')$ is the distance function of the k^{th} image, $x' = x + \Delta_x$ and $y' = y + \Delta_y$ according to the translation $\overline{\mathcal{T}}_k$ estimated for the k^{th} image. According to [MB00], a value for n between 3 and 4 has proven to give good results and the higher the parameter n is, the sharper the transition between the images, and the less blurred the transition regions appear. Concerning the misalignment errors, they can be partly hidden with a small value for n. The figure 6.17 shows the blended image obtained with $n = 4$.

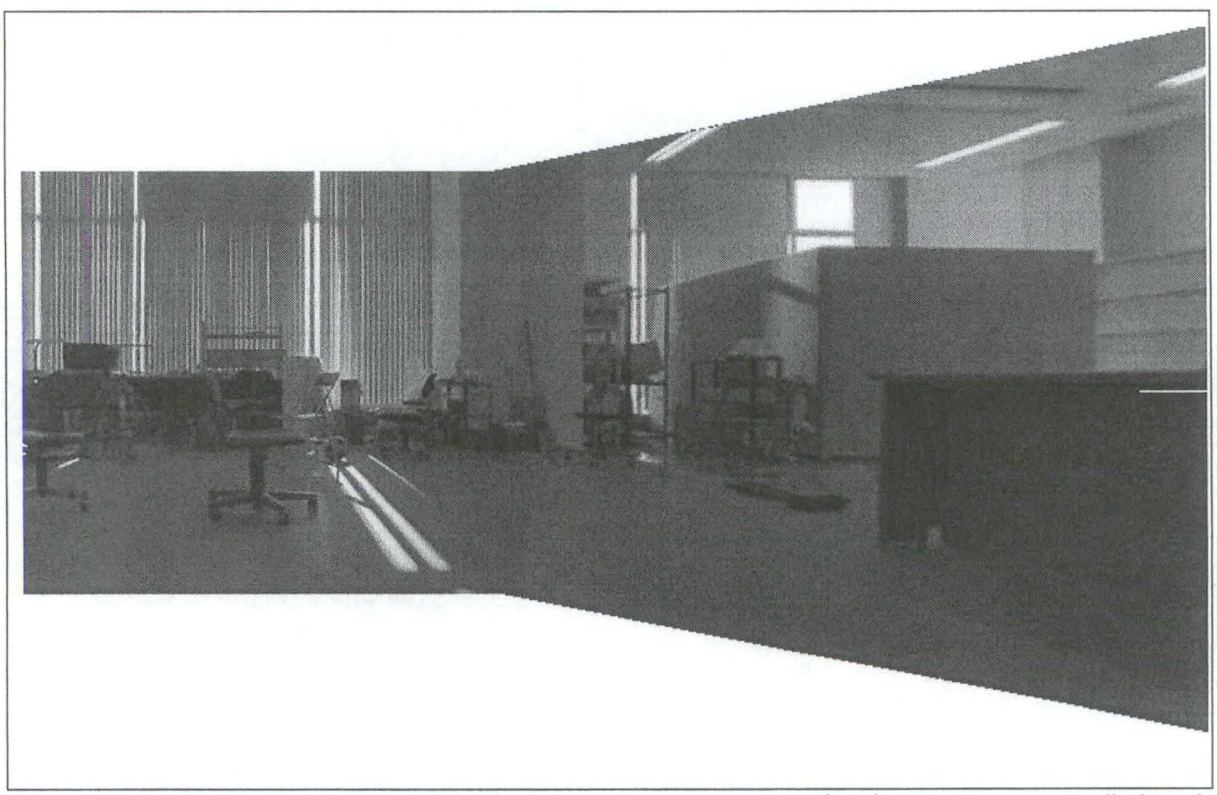


Figure 6.17: The result using the weighted average value as in equation (6.14). The images are well aligned and the misalignment errors are not visible.

6.4 Final result

In this section, we present a large panoramic view composed from 4 colour images (see figure 6.18) and the computation time for each stage (see table 6.3). The photographs are taken with a hand-held camera placed on a basic tripod. A small translation and an approximate rotation of 30 ° are applied in the respective direction and orientation between the poses to simulate the configuration of the future compound eye camera. The focal length is of 7mm but we had to estimate the effective pixel size. The angle of rotation and the translation are also inaccurate because of the material used for snapshots. In addition, the snapshots are resized to 352 × 264 pixels with a standard software for image editing. The pixel size is then estimated to be of 0.0216mm in both horizontal and vertical axis.

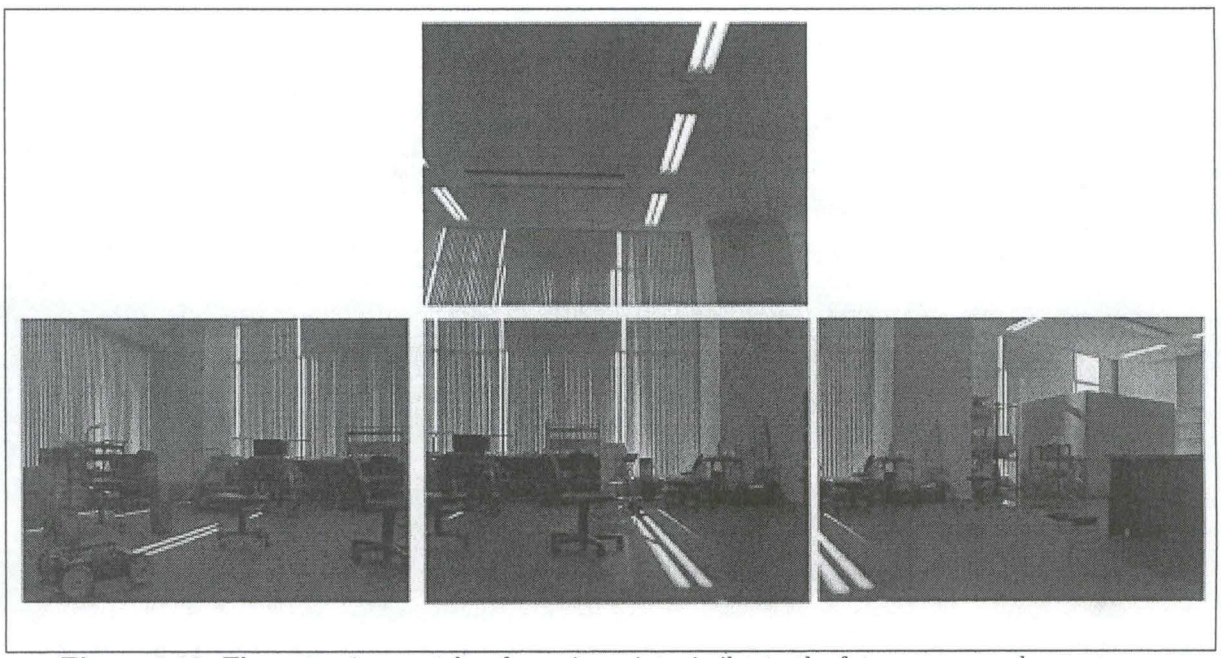


Figure 6.18: The source images taken from viewpoints similar to the future compound eye camera.

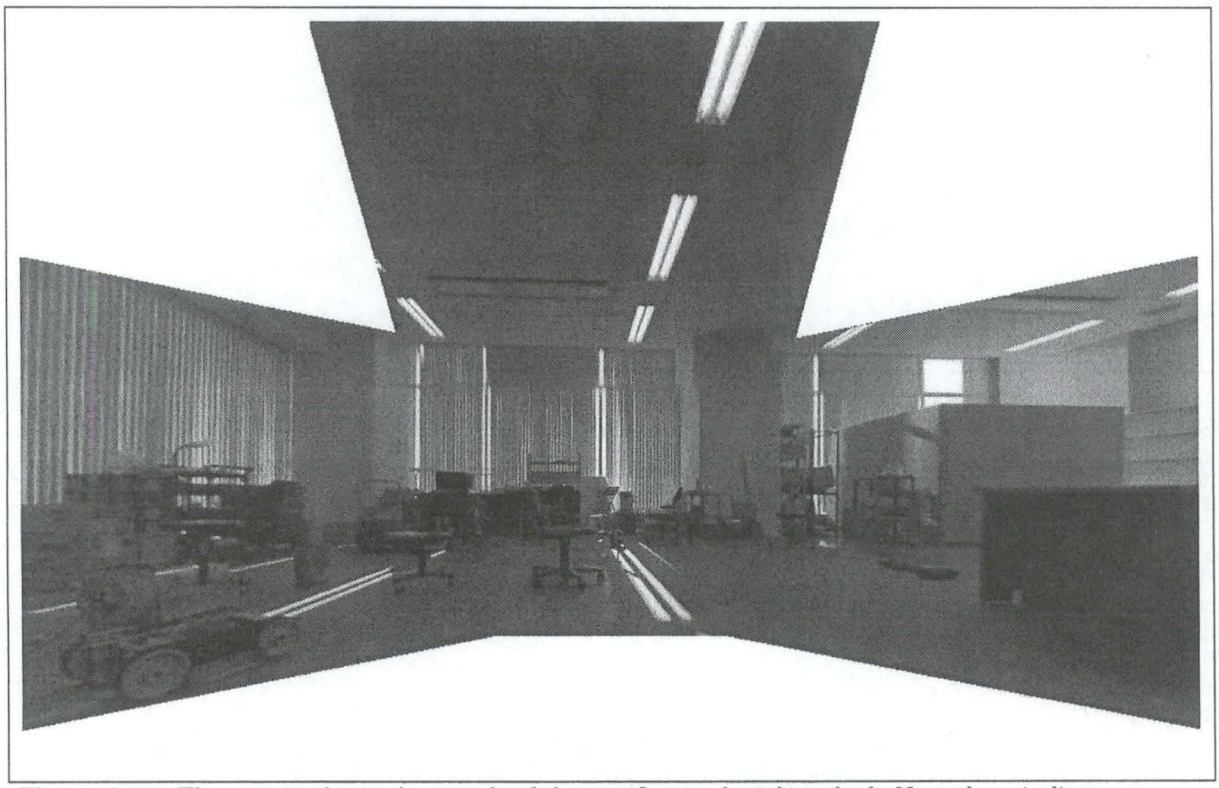


Figure 6.19: The mosaic obtained as result of the rectification-based method. Note the misalignment error around the column between the two windows. This error is due to the inaccuracy of angle of rotation and translation between the poses.

Stage	Time in ms
Rectification	
adjacent up	671ms
adjacent right	451ms
adjacent left	410ms
Rectification total time	*1532ms*
Estimation of translations	
adjacent up	100ms
adjacent right	91ms
adjacent left	110ms
Translation total time	*301ms*
Composition	*2443ms*
Total computation time	**4276ms**

Table 6.3: The computation time of the whole algorithm and its different stages. The computer which has been used is equipped with a processor Intel(R) Pentium(R) 4 2GHz and 256MB DDR of memory. The OS is Microsoft Windows XP. The algorithm is implemented with Java2 SDK Standard Edition v1.4.1 using the integrated development environment Sun ONE Studio 4 CE.

Chapter 7

Depth estimation

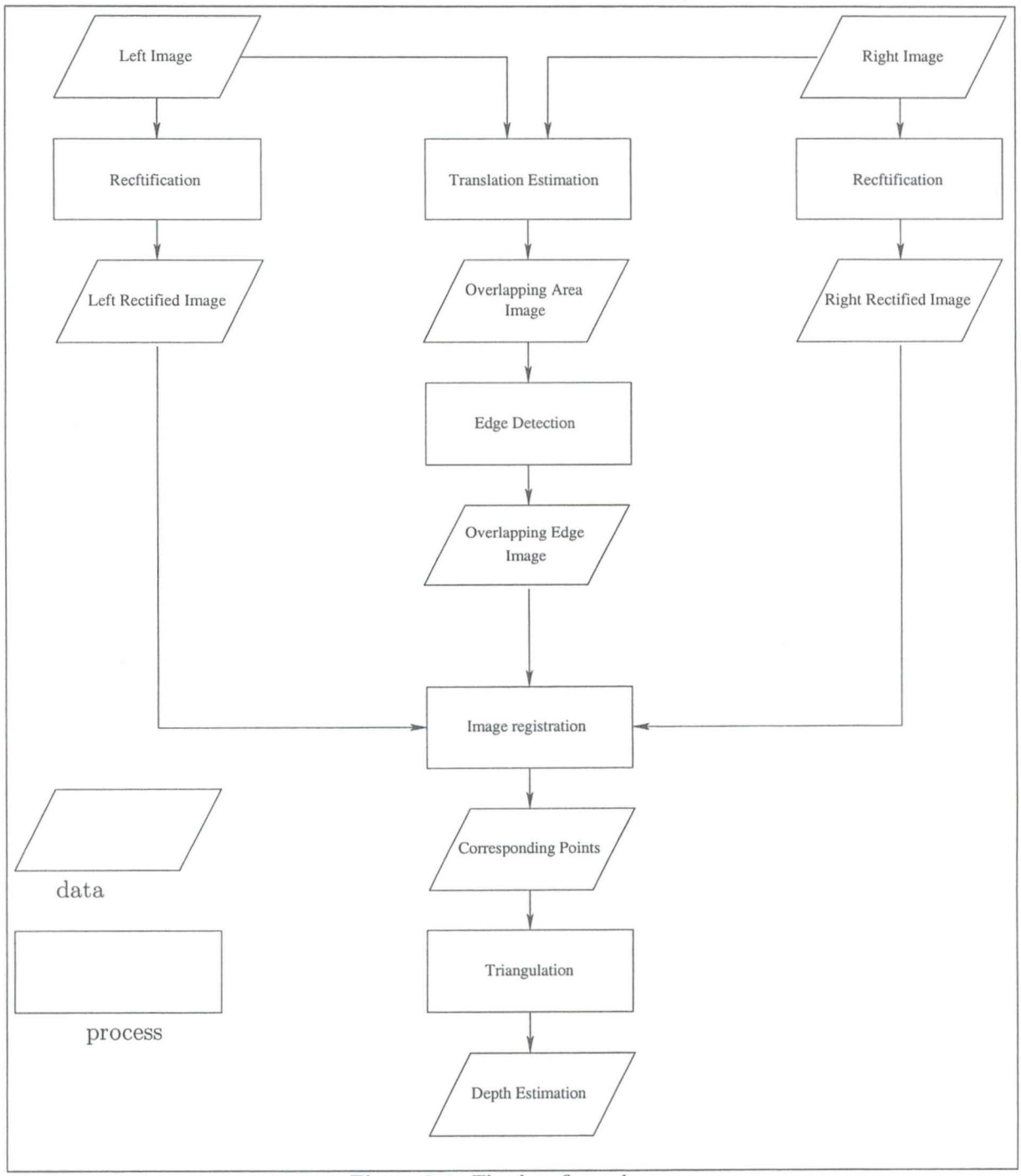


Figure 7.1: The data-flow scheme.

From at least two views of a same scene, the depth estimation of a 3D scene point can be achieved by triangulation under condition that the relative position of the two cameras is known.

extremely difficult task in computer vision. The problem of the corresponding points (called the registration[1] problem) is that from a point in one image, we have to find which point in the second image *corresponds* to the same 3D scene point. Starting from this consideration, we try to develop an algorithm that is able to find the depth of all points visible in the two images and then to make a reconstruction of the 3D environment. It is obvious that the two cameras from which we take the images have to be oriented in such a way that the resulting images are partially overlapping, i.e. there is a part of one image which represents the same scene than a part of the second image. The figure 7.1 shows the scheme of the algorithm we implement.

7.1 Camera settings

First, we will define the extrinsic parameters used for the two cameras. As the module on which lie the cameras is a dodecagon, the angle between them is equal to 30°. Thus the rotation matrix is given by

$$\mathcal{R} = \begin{pmatrix} \cos 30° & 0 & \sin 30° \\ 0 & 1 & 0 \\ -\sin 30° & 0 & \cos 30° \end{pmatrix} \tag{7.1}$$

We choose the world reference frame centred in the left camera centre, this implies that the translation vector of the right camera from the left camera will be (see figure 7.2)

$$\mathcal{T} = \begin{pmatrix} 28 \\ 0 \\ 8 \end{pmatrix} \tag{7.2}$$

expressed in millimeter.

The figure 7.3 shows the original images taken with a system of two cameras parameterised with the previous settings.

7.2 Image rectification

A solution to simplify the search of the corresponding points is image rectification[2]. Due to the rotation and the translation between the two cameras, the corresponding point in the right image (for example) of a point in the left image could be anywhere. Fortunately, the epipolar geometry[3] allows us to search the corresponding point in one image just along the epipolar line generated by the point in the other image. Thus, knowing the epipolar geometry, we have to search only in 1D instead of in 2D.

The idea of using rectification is that we make the two images coplanar and parallel to the baseline. This transformation has the effect that the epipolar lines become horizontal (since the epipoles are at infinity).

[1] See chapter 3 "Image registration methods" for further details.

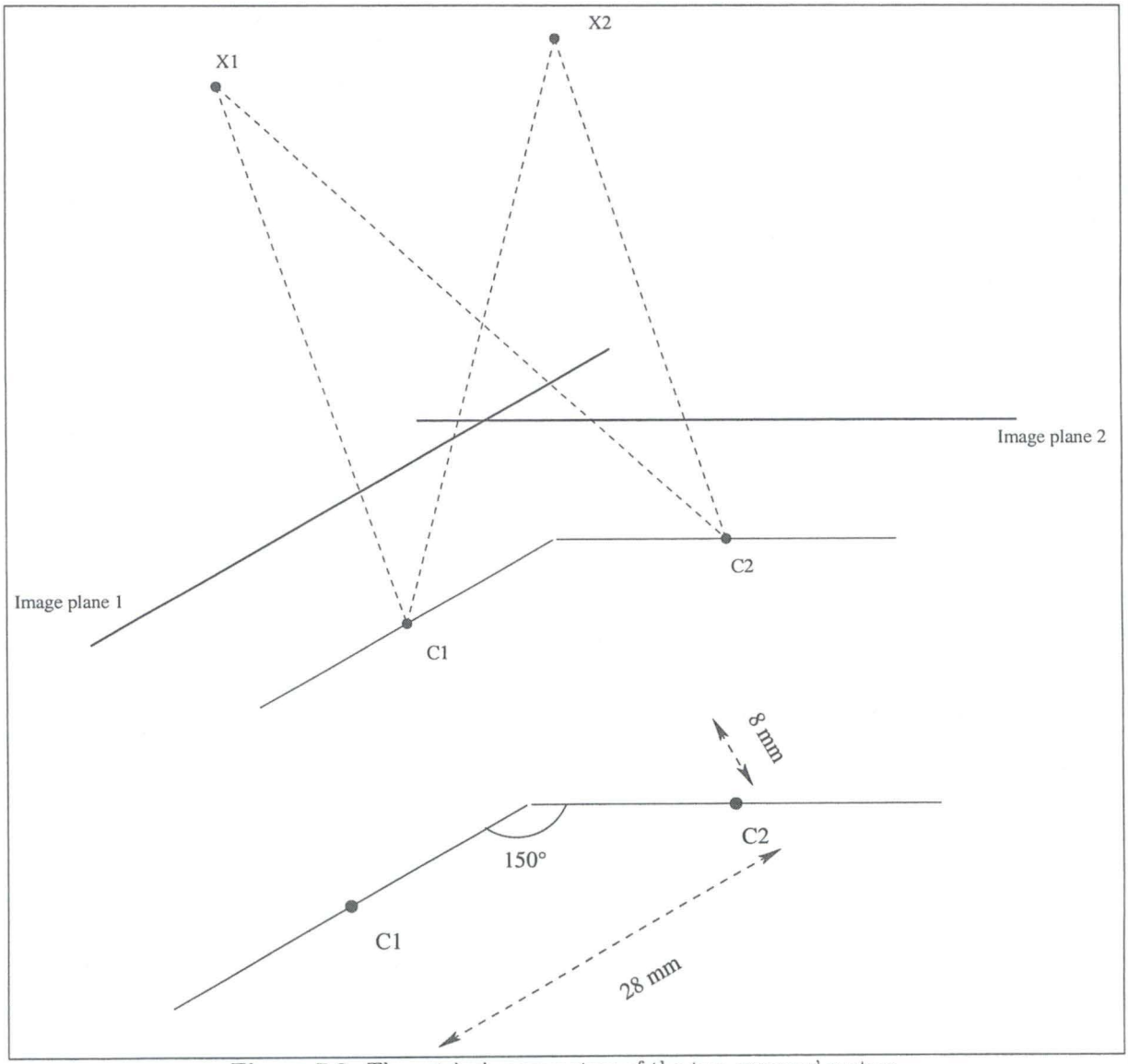


Figure 7.2: The extrinsic parameters of the two cameras' system.

Example :

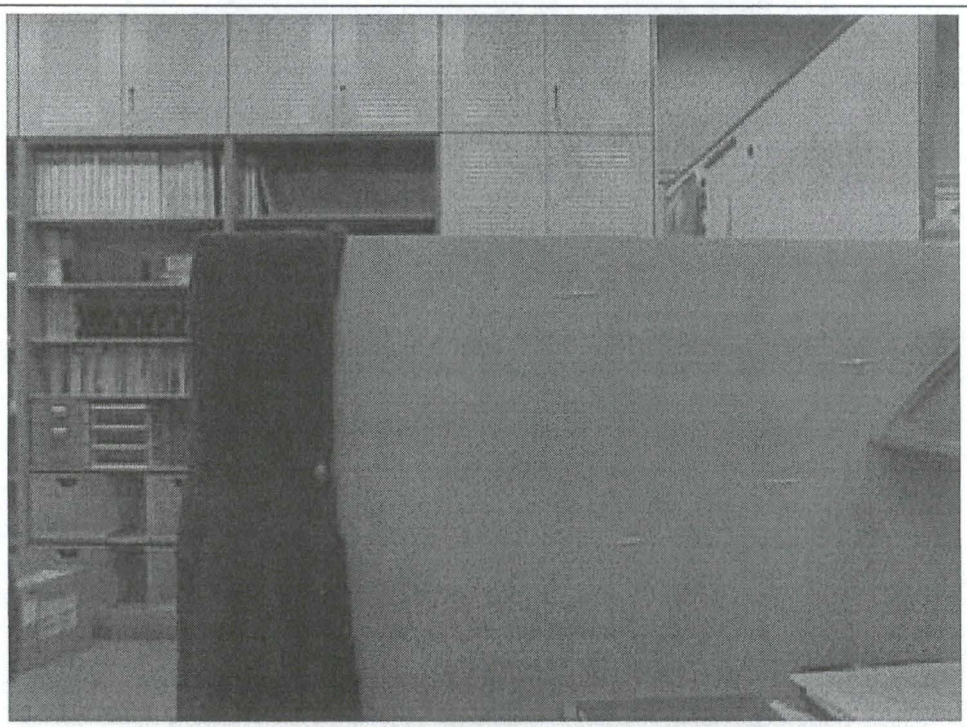
If a point in the left rectified image has the coordinates (in pixel)

$$(x, y)$$

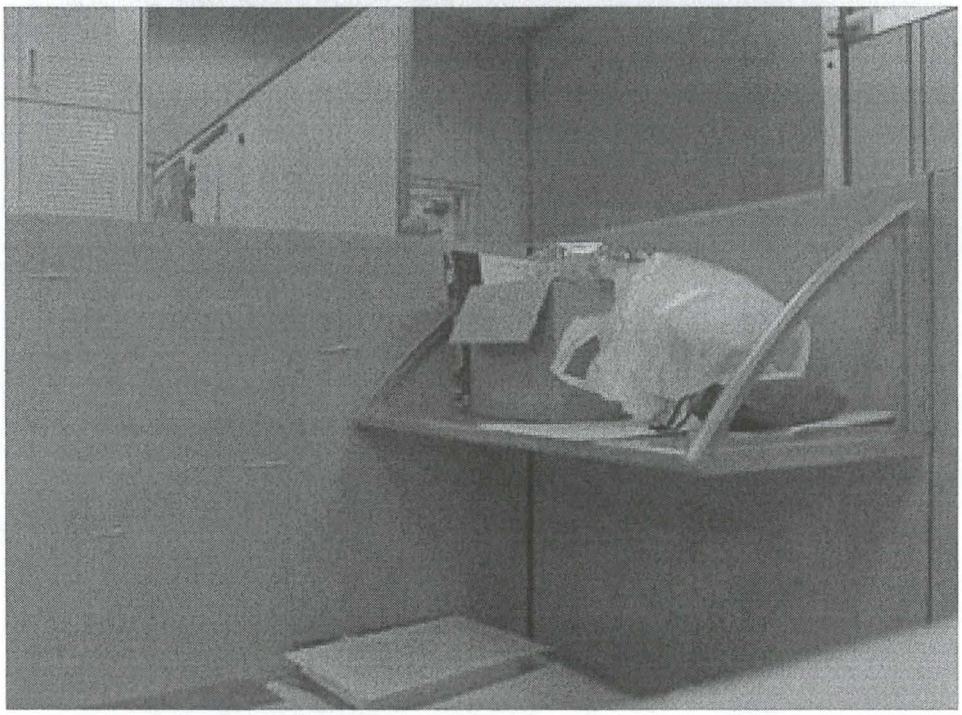
the corresponding point in the right rectified image will be at coordinates

$$(x', y)$$

where we only have to find x'.

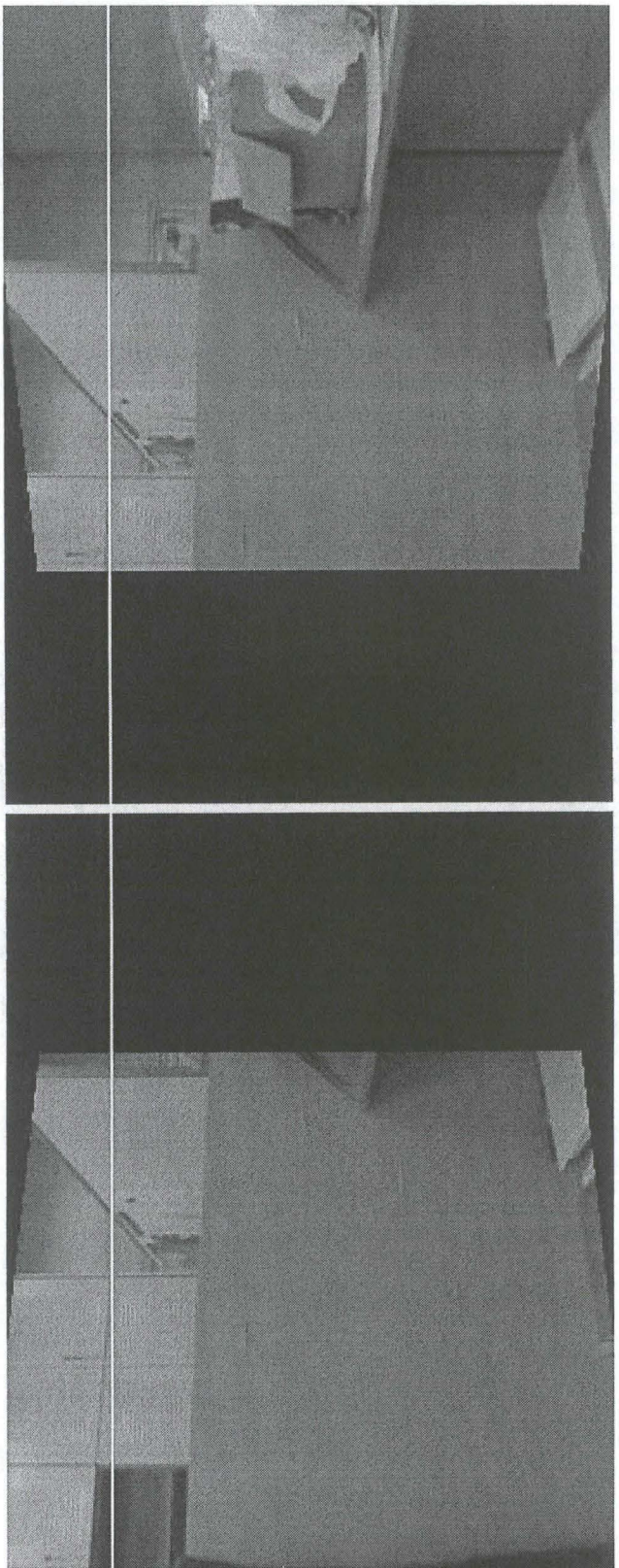


(a)



(b)

Figure 7.3: Figure 7.3(a) is the left original image and figure 7.3(b) is the right original image.



7.3 Interesting points

Naturally, we won't try to compute the corresponding points of the whole image. A part of the left image is not visible in the right image, thus the idea is to search only for points that we are sure to be represented in the right image. Actually it is not interesting to work with the entire image since a lot of information that it contains are useless.

7.3.1 Translation estimation

We try to estimate the translation[4] between the two images, this estimation allows us to compute the *overlapping area*. This area is the part of the left image in which almost all the points are visible in the right image. In other words, almost all points in this area have a corresponding point in the right image. The figure 7.5 shows the overlapping area computed from the images in figure 7.3.

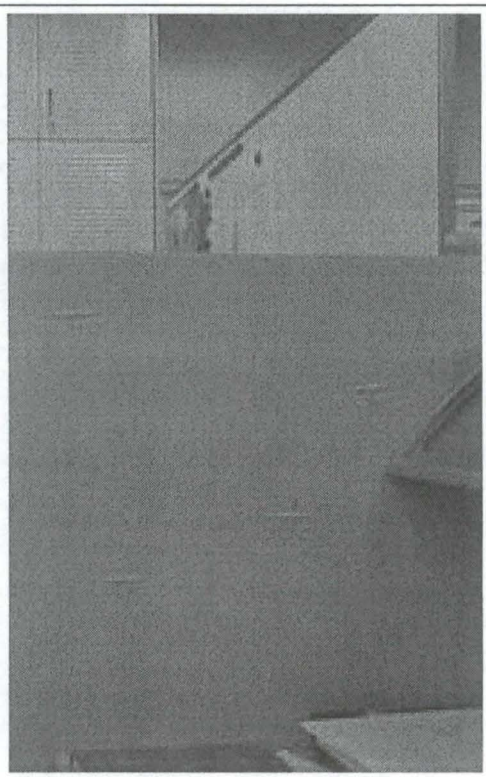


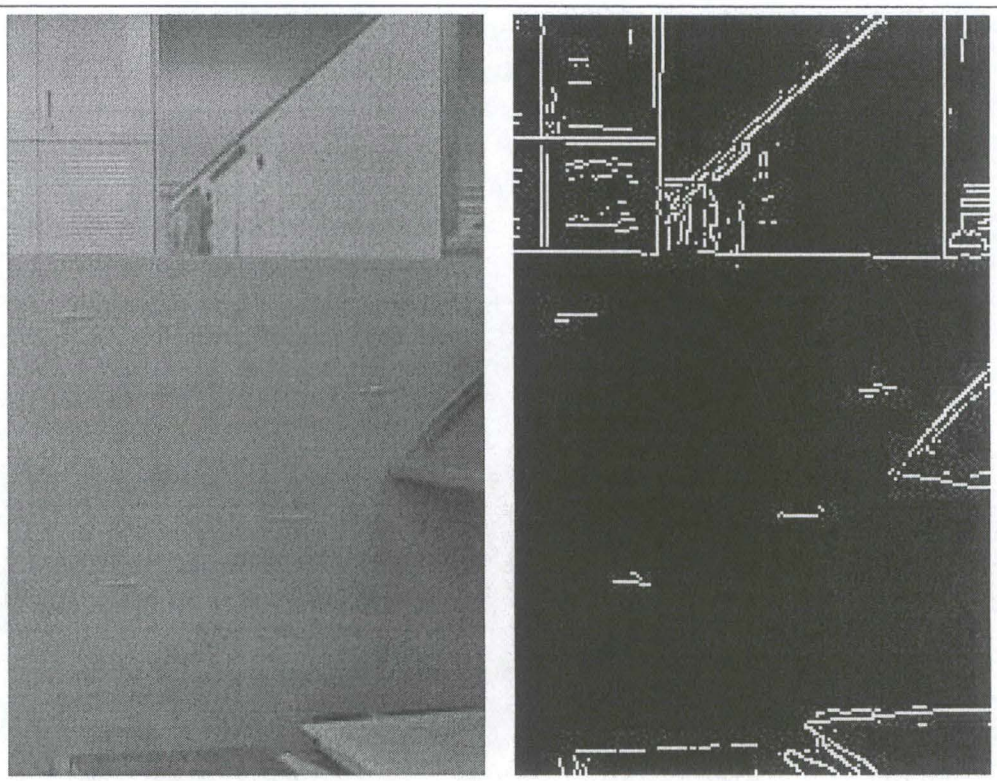
Figure 7.5: The overlapping area from the two original images.

7.3.2 Using edge detection

The translation estimation gives an area on which we can work, but all this area is not useful. We don't need to compute the depth of all the appearing points. Most of the points are useless since they don't provide new interesting information. Actually edge points of the scene are sufficient, computing the depth of these points will give us the distance, the position

and the orientation of the objects of the scene.

We choose the Canny edge detection[5] algorithm which is the commonly used technique. Using this algorithm, the resulting image contains only the useful points for which we need to estimate the depth. Thus the amount of computation to make will largely be reduced. The figure 7.6 gives an example of an edge image.



(a) (b)

Figure 7.6: The overlapping area before (figure 7.6(a)) and after (figure 7.6(b)) the application of the Canny edge detection algorithm.

Notice that the parameters of the algorithm have to be changed if we want to obtain more edges on the image. This is the choice of the user to define when the result is accurate enough.

7.4 Image registration

We previously saw that the image rectification simplifies the registration problem since it allows us to search along an horizontal line. In addition, we know that the coordinates of the corresponding point in the right image are "not far" from the initial coordinates of the point in the left image.

As the figure 7.1 presents it, we will use the two rectified images and the overlapping area to compute the corresponding points. The idea is simple: for each point in the overlapping

edge image we test if it is an edge point (if the point is white), if it is the case we use the rectified images to find its corresponding point.

For facility, we choose a correlation-based method[6] to execute the registration.

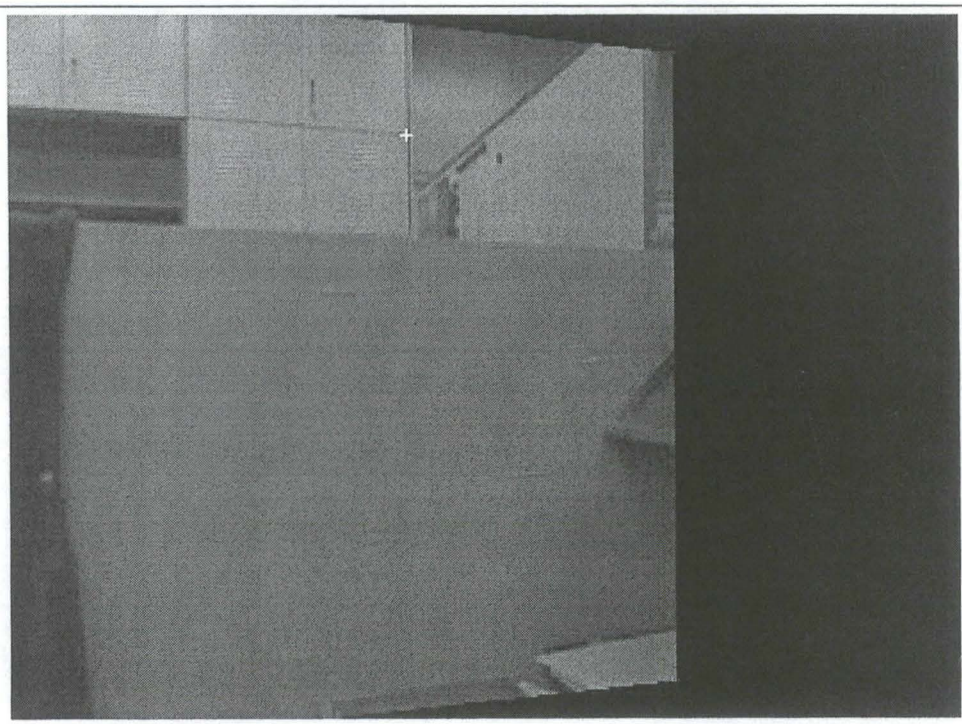


Figure 7.7: The white cross is an arbitrary point for which we want to find the corresponding point.

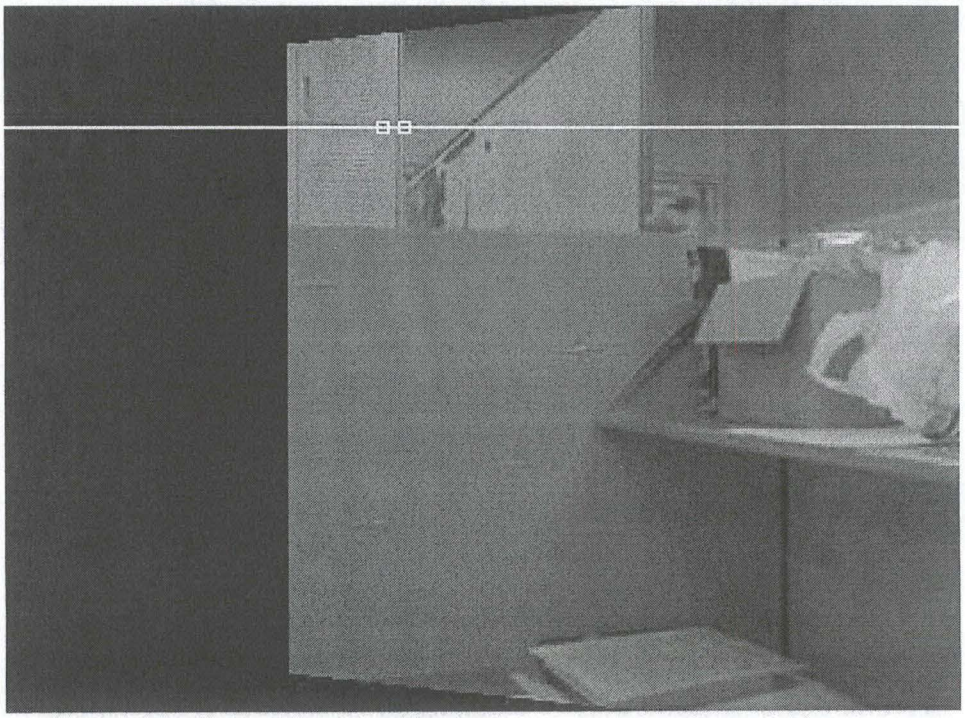


Figure 7.8: Examples of search windows around the initial position and along the horizontal line.

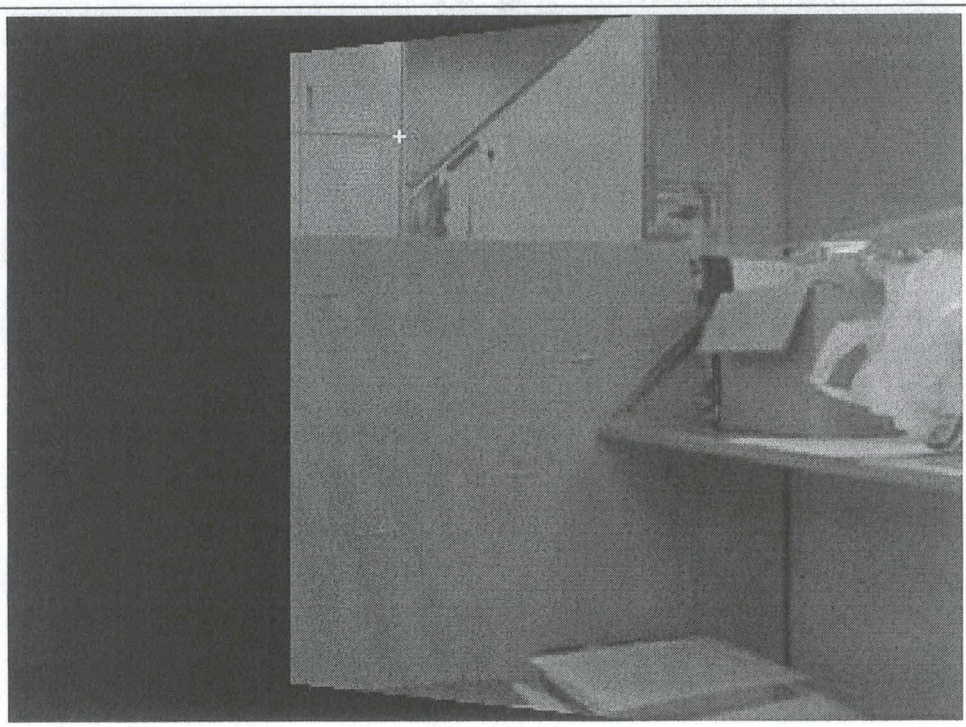


Figure 7.9: The corresponding point found by the correlation-based method.

Applying the correlation-based method to the entire overlapping edge area, we obtain a list of couples of coordinates:

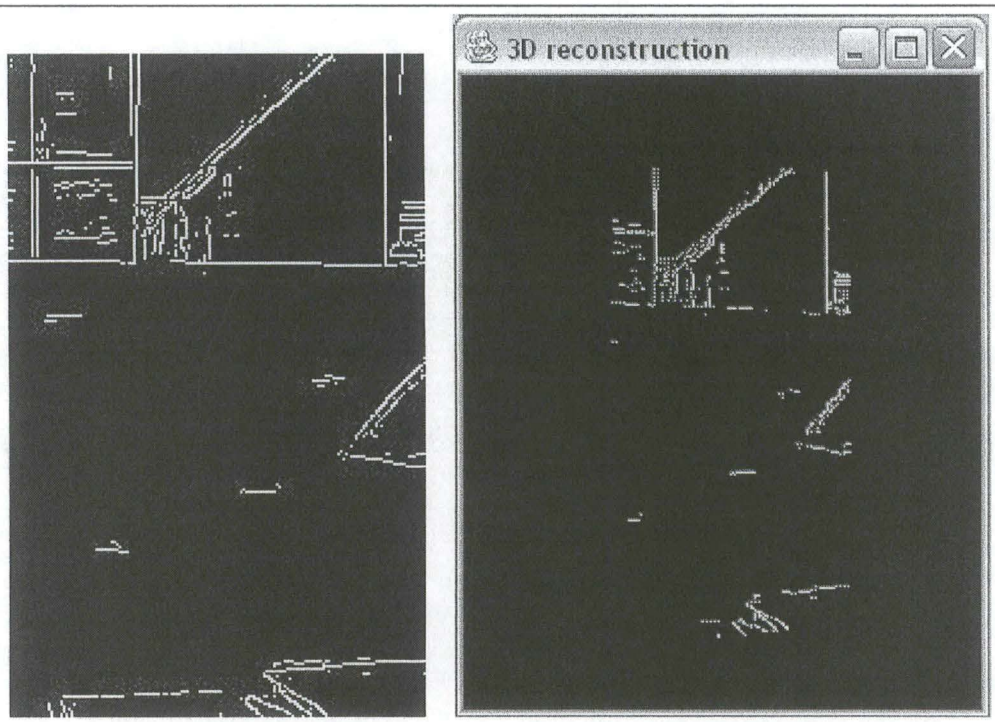
$$[((x_1, y_1), (u_1, v_1)), ((x_2, y_2), (u_2, v_2)), \ldots, ((x_n, y_n), (u_n, v_n))] \tag{7.3}$$

where n is the number of edge points, (x_i, y_i) are the coordinates in the left image and (u_i, v_i) are the coordinates of the corresponding point.

7.5 Triangulation

From the list of couples of corresponding points, we use the mid-point triangulation technique to obtain a depth estimation for each couple of coordinates. The result is a list of 3D coordinates

$$[(X_i, Y_i, Z_i) | i = 1, \ldots, n] \tag{7.4}$$



(a) The edges in the overlapping area (b) The 3D reconstruction of the overlapping area

Figure 7.10: The 3D reconstruction rendered with Java3D v 1.3.1.

Figure 7.10 shows a example of the 3D reconstruction of the scene from the common vision of the two images. However, a lot of points are badly positioned or missing. We think this is principally due to the lack of precision of the intrinsic and extrinsic parameters of the images used for the test. Indeed, the stereo system simulated with the hand-held camera is very inaccurate. Test should be done with the final prototype to completely know the result of the proposed approach.

Conclusion

Conclusion

Concerning the panoramic image generation, we have presented a rectification-based approach that works in three main stages:

1. *rectification* of the images in order to reduce the alignment problem to a simple translation;
2. *estimation of the translation* based on a coarse grid and an exhaustive search between the reference image and each rectified image;
3. *composition* of the reference image and the rectified images into a seamless panoramic image using a blending algorithm.

This approach in image mosaic techniques reduces the search of the transformation between images to a simple translation. Thus, complex registration stages are not required and it should provide better computation time. It has also the advantage of handling large rotations between viewpoints and not requiring a large overlapping area between images. However, there are a few limitations of our method. A reference image is necessary among the images to be aligned and must have a central position so that the other images are adjacent to this reference image. In addition, it doesn't provide accurate alignment of images despite satisfactory visual results. The misalignment errors are not corrected but disguised thanks to the blending algorithm used in the last stage. This is a reasonable choice between computation time and quality of results in our case because the source images come from CMOS cameras.

The depth estimation method presented in this work have two main advantages:

1. the rectification approach simplifies the registration problem;
2. all techniques used are easy to understand.

Unfortunately, it couldn't be tested with precision because of the lack of available images. Nevertheless, we can already see some failures which will cause bad results.

First, the choice of the correlation-based technique to solve the registration problem is not necessary the best. Even if this technique is simple to use, the weaknesses exist. For example:

1. the conditions of enlightenment will greatly affect the result;
2. surfaces with few variations of intensity (such walls, panels, ...) are not treated properly.

Another problem comes from the rectification itself. The interpolation used introduces imprecision and loss of information. Indeed the image transformation deforms it and may be not accurate if the parameters are not known exactly. This will cause a lack of precision.

The impact of these drawbacks will be wrong correspondences and of course big mistakes in depth estimation since the parameters of the triangulation are false. Furthermore, we

The described panoramic image generation technique has worked well in the scene in which we tried it and using a hand-held camera to take the pictures. However, the depth estimation is very sensitive to inaccurate parameters (i.e. intrinsic and extrinsic parameters). Thus, intensive experiments with accurate parameters and improvements are necessary in order to apply it to the real application and to validate the two methods:

- tests with pictures from the compound eye camera and under conditions similar to rescue activities (dark scene with debris) must be done as soon as the prototype is ready;
- the complexity of the panoramic image generation has to be reduced, mainly in the composition stage, in order to run in real-time.

We trust in the successful use of the rectification approach in the project "An Intelligent Sensor Head for Information Collection in Debris". On the one hand, the proposed methods should work well with dark pictures of debris because the only critical part is the estimation of translations which makes optimal use of all the information and is improved thanks to the rectification stage. On the other hand, the complexity of the panoramic image generation algorithm should be easily improved because the algorithm was implemented in a simple way in order to demonstrate the applicability of the rectification-based approach.

An idea to overcome the problem introduced by the rectification used in depth estimation could be to directly use the epipolar lines equation. The search of the corresponding points would be along these lines without modifying the images. The price to pay will be a slower technique and then a loss of speed in the processing of the images.

Finally and for all these reasons, it appears clearly that ours algorithms are the first step on the way to an optimal solution. In addition, further works may consist of:

- trying other faster techniques;
- estimating the depth with more than two source images;
- building 3D maps of the scene;
- registering the current images with images taken during a former passage or by other robots (e.g. multi-agents);
- building a map in order to compute the path of robots;
- automatically detecting victims;
- building a model of debris to estimate their stability and the possible danger allowing to plan their removal.

Bibliography

[Aga02] Gady Agam. CS511 - Fall 2002 - Lecture 9 Notes. `http://www.csam.iit.edu/~agam/cs511/lect-notes/lect9/lect9.html`, 2002. (Date of last change 02/10/2002) (Date of access 14/08/2003).

[Bro92] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys*, volume 24(number 4):pages 325–376, January 12 1992.

[Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 8(number 6):pages 679–698, November 1986.

[CL99] L.C. Chen and J.D. Lee. Initialization for Image Registration using Feature Matching. `http://www.gisdevelopment.net/aars/acrs/1999/ps1/ps1068.htm`, 1999. (Date of access 05/05/2003).

[Der98] F. Deravi. Frequency Domain - The 2D Fourier Transform. `http://visl.technion.ac.il/labs/anat/index2.html`, 1998. (Date of last change 15/01/1998) (Date of access 16/08/2003).

[Fau93] Olivier Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint.* The MIT Press, Cambridge, MA, November 19 1993.

[Fis02] Robert Fisher. CVonline: Vision Geometry and Mathematics. `http://www.dai.ed.ac.uk/CVonline/geom.htm`, 2002. (Date of last change 18/07/2003) (Date of access 14/08/2003).

[FL95] Olivier Faugeras and Quang-Tuan Luong. The fundamental matrix: theory, algorithms and stability analysis. *The International Journal of Computer Vision*, volume 17(number 1):pages 43–76, 1995.

[FLP01] Olivier Faugeras, Quang-Tuan Luong, and Théodore H. Papadopoulo. *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications.* The MIT Press, Cambridge, MA, March 5 2001.

[FPWW02] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. The Hypermedia Image Processing Reference. `http://www.dai.ed.ac.uk/HIPR2/index.htm`, 2002. (Date of last change October 2002) (Date of access 15/08/2003).

[HB00] Chiou-Ting Hsu and Rob A. Beuker. Multiresolution feature-based image registration. In *Visual Communications and Image Processing 2000*, volume 4067, pages 1490–1498, Perth, Australia, June 20-23 2000.

[HE] Ugur Halici and Ali Erol. MET_U 903 Image Processing Algorithms. `http://www.ii.metu.edu.tr/metuonline/courses/met_u903/html/master/lectu%res/index.html`. Lectures notes from METU Informatics Institute. (Date of access 25/04/2003).

[HLF+97] Jun-Wei Hsieh, Hong-Yuan Mark Liao, Kuo-Chin Fan, Ming-Tat Ko, and Yi-Ping Hung. Image registration using a new edge-based approach. *Computer Vision and Image Understanding: CVIU*, volume 67(number 2):pages 112–130, 1997.

[Ioc98] Luca Iocchi. Stereo Vision: Triangulation. `http://www.dis.uniroma.it/~iocchi/stereo.triang.html`, 1998. (Date of last change April 6 1998) (Date of access 17/04/2003).

[JC01] Yao Jianchao and Chia Tien Chern. The practice of automatic satellite image registration. In *22nd Asian Conference on Remote Sensing*, volume 1, pages 221–226, Singapore, November 5-9 2001.

[Leh] Steven Lehar. An Intuitive Explanation of Fourier Theory. `http://cns-alumni.bu.edu/~slehar/fourier/fourier.html`. (Date of access 05/05/2003).

[Lev95] Silvio Levy. The Geometry Center Home Page. `http://www.geom.uiuc.edu/docs/reference/CRC-formulas/book.html`, 1995. Excerpted from the 30th Edition of the *CRC Standard Mathematical Tables and Formulas* published in late 1995 by CRC Press. (Date of last change 04/10/1995) (Date of access 14/08/2003).

[Lew95] J.P. Lewis. Fast normalized cross-correlation (expanded version of the paper). *Vison Interface*, pages 120–123, 1995.

[Lia00] Yi Liang. Phase Correlation Motion Estimation. `http://ise.stanford.edu/class/ee392j/projects`, 2000. EE392J Digital Video Processing. (Date of last change 11/03/2000) (Date of access 01/05/2003).

[MB00] Laurent Meunier and Moritz Borgmann. High-resolution panoramas using image mosaicing. Technical report, Stanford University, May 2000. EE368 Digital Image Processing.

[McG98] Morgan McGuire. An image registration technique for recovering rotation, scale and translation parameters. Technical report, NEC Research Institute, Princeton, NJ, February 19 1998.

[MNLM99] David M. Mount, Nathan S. Netanyahu, and Jacqueline Le Moigne. Efficient algorithms for robust feature matching. *Pattern Recognition*, volume 32(number 1):pages 17–38, 1999.

[RC96] B.S. Reddy and B.N. Chatterji. An FFT-based technique for translation, rotation,

[Rod] Lawrence H. Rodrigues. What is interpolation? `http://www.awprofessional.com/`. Addison-Wesley Professional homepage. (Date of access 20/08/2003).

[SC94] Richard Szeliski and James Coughlan. Spline-based image registration. Technical Report number 94/1, Digital Equipment Corporation, Cambridge Research Lab, April 1994.

[SS97] Heung-Yeung Shum and Richard Szeliski. Panoramic image mosaics. Technical Report number MSR-TR-97-23, Microsoft Research, Redmond, WA, 1997.

[Sze96] Richard Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, volume 16(number 2):pages 22–30, March 1996.

[Tao02] Hai Tao. Image Features: Edges and Corners. `http://www.cse.ucsc.edu/classes/cmpe264/Spring2002/`, 2002. Lecture notes CMPE 264: Image Analysis and Computer Vision. (Date of access 05/05/2003).

[TH98] Miroslav Trajković and Mark Hedley. Fast corner detection (extended version). *Image and Vision Computing*, volume 16(number 2):pages 75–87, 1998.

[TKSW03] Satoshi Tadokoro, Takumi Kishima, Julien Stouffs, and Yannick Willame. An intelligent sensor head for information collection in debris. In *11th International Conference on Advanced Robotics (ICAR 2003)*, University of Coimbra, Portugal, June 2003.

[TV98] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3D Computer Vision.* Prentice Hall, Englewood Cliffs, NJ, March 6 1998.

[WYZ01] Chun Kit Wilbur Wong, Ruiduo Yang, and Yan Zhang. Image mosaics under simple planar scenes and its application. Technical report, Hong Kong University og Science and Technology, Kowloon, Hong Kong, December 2001.

[ZB01] Zhong Zhang and Rick S. Blum. A hybrid image registration technique for digital camera image fusion application. *Information Fusion*, volume 2(number 2):pages 135–149, 2001.

[ZT98] Djemel Ziou and Salvatore Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, volume 8:pages 537–559, 1998.

APPENDIX

A-1 The interpolation

The image pixels occupy integer coordinates x and y and so the intensity function $I(x, y)$ usually has the signature $I : \mathbb{N}^+ \times \mathbb{N}^+ \rightarrow \mathbb{N}^+$. Sometimes however, we want to get the brightness of a pixel $P_{(x',y')}$ from float indices x' and y' i.e. subpixel intensity. In this case we have $I : \mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \mathbb{N}^+$. Thus, we interpolate the pixel intensity i.e. the value of the pixel is generated from its neighbours. The contribution of neighbouring pixels is often related to a certain weight which is generally inversely proportional to the distance at which the neighbour is located.

In this section, we briefly present the most common types of interpolation in two dimension according to [Rod].

A-1.1 The nearest-neighbour interpolation

One can simply round x and y to get the intensity from these rounded coordinates i.e. the interpolating pixel is assigned the value of the nearest neighbour. This method is fast, but it is the simplest and crudest technique and it may not produce accurate results. Indeed, the nearest-neighbour interpolation may introduce aliasing[8] ,1/2 pixel shifting and *jaggies*[9].

A-1.2 The bilinear interpolation

In linear interpolation, the value of the interpolated pixel $P_{x'}$ is computed from the immediate neighbours of $P_{x'}$. Let $P_{x'}$ be the pixel to be interpolated where $x' = x + d$ with $x \in \mathbb{N}^+$ and $0.0 \leq d < 1.0$. The distance-to-weight relationship is linear i.e. the relationship is of the form $y = ax + b$ as shown on figure A-1.

Thus in 2D, we have the bilinear interpolation, also known as first-order interpolation, which linearly combines the values of the four closest pixels i.e. in a 2×2 pixel neighbourhood. First we apply the linear interpolation in one direction. Then the linear interpolation in the other direction is applied on the result. Let the non-integer coordinates (x', y') be $(x + d_x, y + d_y)$ with $x, y \in \mathbb{N}^+$ and $0.0 \leq d_x, d_y < 1.0$. Consequently, $P_{(x,y)}$ denotes the upper left-hand neighbour (see figure A-2).

The bilinear interpolation is computed by:

$$I(x', y') = (1-d_x)(1-d_y)I(x, y)+d_x(1-d_y)I(x+1, y)+d_x d_y I(x+1, y+1)+d_y(1-d_x)I(x, y+1) \quad \text{(A-1)}$$

The bilinear interpolation is piecewise bilinear and provides an improvement in image quality over the nearest-neighbour interpolation. However, it may result in less-than-desirable smoothing effects.

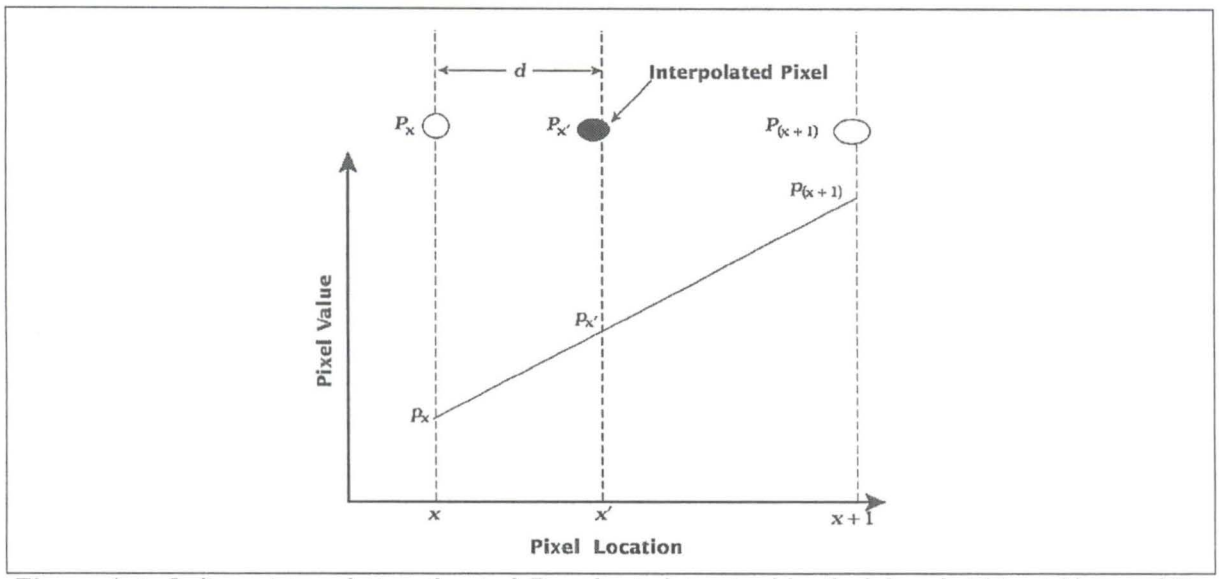


Figure A-1: In linear interpolation, the pixel $P_{x'}$ value is determined by the left and right neighbours of $P_{x'}$ i.e. P_x and $P_{(x+1)}$. Their respective values are noted p_x and $p_{(x+1)}$ The intensity $p_{x'}$ of $P_{x'}$ is a linear function of p_x and $p_{(x+1)}$ as it is shown by the drawn line.

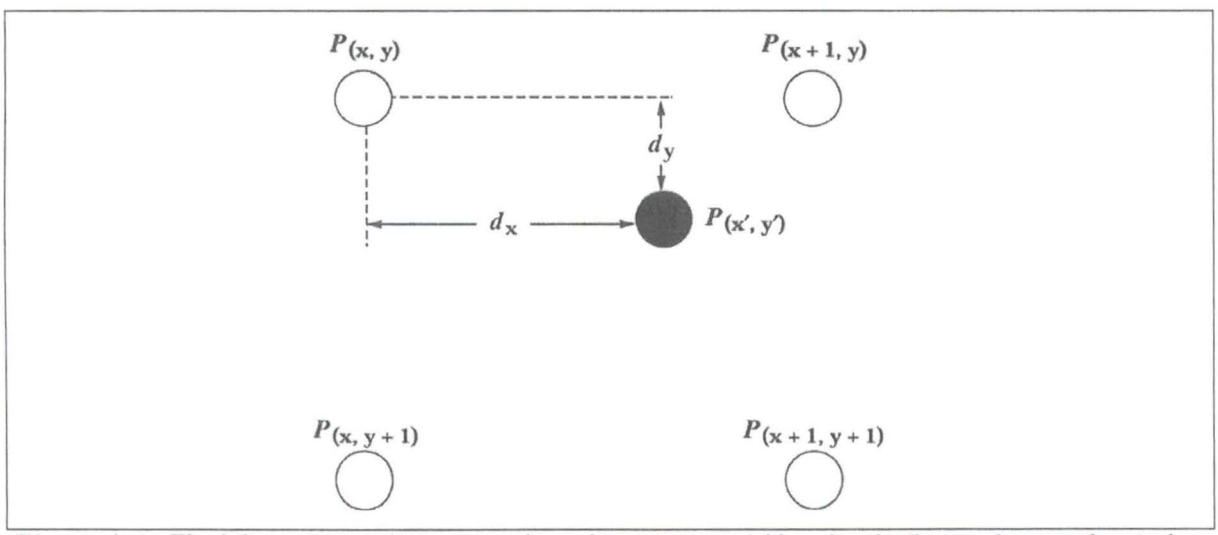


Figure A-2: The bilinear interpolation is performed in a 2×2 neighbourhood. $P_{(x',y')}$ denotes the pixel at distance d_x and d_y from the upper left-hand neighbour $P_{(x,y)}$.